Design and analysis of models and algorithms for optimisation in networks

Thesis submitted in order to obtain the degree of Doctor in Applied Economics

Author: Jochen JANSSENS

Promotor: Prof. dr. Kenneth Sörensen

May, 2016
Promotor
Prof. dr. Kenneth Sörensen (University of Antwerp, Belgium)

Members of the Examination Committee
Prof. dr. Trijntje CornelisSENS (University of Antwerp, Belgium)
Prof. dr. Pieter VanstEEEnWegen (KU Leuven, Belgium)
Prof. dr. Genserik Reniers (University of Antwerp, Belgium)
Prof. dr. Marc SevaUX (University of South Brittany, France)
Prof. dr. An Caris (University of Hasselt)
Acknowledgements

Along the past few years, many people have been very supportive and helpful in the process of writing my doctoral thesis. While probably forgetting many, I would like to thank the most important ones.

First and foremost, I would like to thank my supervisor Kenneth Sörensen, without whom this work would not have been possible. I would like to thank him for all his guidance and advice, for teaching me how to do research and guide me through the process of pursuing a doctoral degree.

I would also like to express my gratefulness to the members of my Examination Committee, An Caris, Pieter Vansteenwegen, Genserik Reniers, Marc Sevaux and Trijntje Cornelissens for proofreading my thesis and all the helpful comments they gave to increase the quality of this manuscript.

I would like to express my enormous gratitude to my friend and colleague, Luca Talarico, for all his advice during the past years. I really loved the chats and discussions we had, and the insights I got from working with such a great researcher.
Acknowledgements

Furthermore, I would like to express my thankfulness to Marco Castro for helping me with all my coding problems, to Daniel Palhazi Cuervo for his expertise in statistics and all the colleagues from the fifth floor for the great conversations during the coffee breaks, the fun weekends and great moments we had together.

Finally, my sincerest gratitude goes to my wife, Jolien, my friends and my family for their friendship and the interest they took in my work, for their support during the past five years, and for asking me time and time again what my research was about.

Jochen Janssens
Antwerp, May 2016
# Contents

<table>
<thead>
<tr>
<th>Acknowledgements</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>xvii</td>
</tr>
<tr>
<td>Samenvatting</td>
<td>xxiii</td>
</tr>
</tbody>
</table>

## NETWORK SECURITY OPTIMISATION

1. Introduction  3
2. Single source and destination security strategy optimisation  9
   2.1 Introduction  11
   2.2 Literature review  12
   2.3 Problem description  18
      2.3.1 Illustrative Example  20
      2.3.2 Mathematical model  27
   2.4 Solution approach  28
      2.4.1 Constructive phase  30
      2.4.2 Improvement phase  33
      2.4.3 Diversification phase  35
   2.5 Computational experiments  36
      2.5.1 Instance generation  36
## Contents

2.5.2 Parameter tuning ................................................. 40  
2.5.3 Computational results ................................. 42  
2.6 Alternative model .............................................. 46  
2.7 Conclusion ......................................................... 50  

3 Security strategy optimisation in networks with multiple source and destination nodes 53  
3.1 Introduction ......................................................... 55  
3.2 Problem description ........................................... 57  
3.2.1 Edge and node failures ................................. 57  
3.2.2 Mathematical formulation ......................... 59  
3.3 Solution approach ............................................... 62  
3.3.1 Auxiliary calculations ................................. 62  
3.3.2 Exact approach ........................................... 64  
3.3.3 Heuristic approach ................................. 65  
3.4 Computational experiments ................................. 72  
3.4.1 Calibration of the metaheuristic ............... 74  
3.4.2 Results ......................................................... 76  
3.5 Conclusion and discussion .................................. 78  

4 Mitigating domino effects by allocating protective safety barriers 81  
4.1 Introduction ......................................................... 83  
4.2 Problem description ........................................... 88  
4.3 Solution approach ............................................... 95  
4.4 Case study ........................................................... 99  
4.4.1 Test instance ............................................. 99  
4.4.2 Results ......................................................... 102  
4.5 Conclusions ........................................................... 109
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Water distribution network design optimisation</td>
<td>113</td>
</tr>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>115</td>
</tr>
<tr>
<td>5.2</td>
<td>Problem definition</td>
<td>116</td>
</tr>
<tr>
<td>5.3</td>
<td>Literature review</td>
<td>118</td>
</tr>
<tr>
<td>5.4</td>
<td>Solution approach</td>
<td>122</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Exact approach</td>
<td>123</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Adaptive large neighbourhood search</td>
<td>129</td>
</tr>
<tr>
<td>5.4.3</td>
<td>Neighbourhoods</td>
<td>133</td>
</tr>
<tr>
<td>5.5</td>
<td>Computational experiments</td>
<td>137</td>
</tr>
<tr>
<td>5.5.1</td>
<td>Instance generation</td>
<td>137</td>
</tr>
<tr>
<td>5.5.2</td>
<td>Statistical analysis</td>
<td>140</td>
</tr>
<tr>
<td>5.6</td>
<td>Conclusion and discussion</td>
<td>145</td>
</tr>
<tr>
<td>6</td>
<td>Routing for Couriers</td>
<td>147</td>
</tr>
<tr>
<td>7</td>
<td>Multi-objective microzone-based vehicle routing for courier companies</td>
<td>153</td>
</tr>
<tr>
<td>7.1</td>
<td>Introduction</td>
<td>155</td>
</tr>
<tr>
<td>7.2</td>
<td>Literature review</td>
<td>158</td>
</tr>
<tr>
<td>7.3</td>
<td>Problem definition</td>
<td>163</td>
</tr>
<tr>
<td>7.4</td>
<td>Methodology</td>
<td>166</td>
</tr>
<tr>
<td>7.4.1</td>
<td>Neighbourhoods</td>
<td>167</td>
</tr>
<tr>
<td>7.4.2</td>
<td>Variable neighbourhood tabu search</td>
<td>171</td>
</tr>
<tr>
<td>7.5</td>
<td>Computational results</td>
<td>173</td>
</tr>
<tr>
<td>7.5.1</td>
<td>Test instances</td>
<td>174</td>
</tr>
<tr>
<td>7.5.2</td>
<td>Results</td>
<td>176</td>
</tr>
<tr>
<td>7.6</td>
<td>Conclusions and future research</td>
<td>182</td>
</tr>
</tbody>
</table>
8 The tuning of parameters of a multi-objective algorithm using PROMETHEE 187
8.1 Introduction .................................. 189
8.2 Literature ..................................... 190
8.3 Metrics for the evaluation of the quality of Pareto sets 194
  8.3.1 Set Coverage ................................. 194
  8.3.2 Hypervolume ................................. 194
  8.3.3 Spacing ....................................... 196
8.4 The PROMETHEE method ......................... 196
  8.4.1 PROMETHEE properties .................... 197
  8.4.2 PROMETHEE algorithm .................... 199
8.5 The PROMETHEE GDSS Procedure ................... 200
8.6 Test instances .................................. 201
8.7 Algorithmic parameter values ..................... 203
8.8 Analysis of the results .......................... 204
8.9 Conclusions .................................... 212
9 Conclusions .................................... 215
Bibliography ..................................... 221
Notation ......................................... 239
Index ............................................. 245
List of Figures

2.1 Utility network $G$ with a source $o$ and a destination $d$.  
2.2 Relationship between the number of edges (a) and critical scenarios (b) and CPU time needed to update the objective function after a move is executed.  
2.3 Example graph $G$.  
2.4 Plot of average objective values and times for the given parameter settings.  
2.5 Plot of the objective value over time.  
2.6 Graphs with different topology.  
3.1 Substitution of a failure of a node (a) with the failure of edges (b) for the computation of the connectivity index.  
3.2 Relationship between the number of edges and the CPU time needed for the exact algorithm.  
3.3 Marking disconnected groups for calculating the connectivity index.  
3.4 Plot of average objective values and computing times for given parameter setting.  
3.5 Plot of the objective value over time.
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Situation I and Situation II in terms of cardinality</td>
<td>89</td>
</tr>
<tr>
<td>4.2</td>
<td>Network scheme overlaying an image retrieved from Google maps showing a park of chemical storage tanks</td>
<td>100</td>
</tr>
<tr>
<td>4.3</td>
<td>Allocation of barriers for different values of $D$.</td>
<td>105</td>
</tr>
<tr>
<td>4.4</td>
<td>Allocation of barriers on each installation for different values of $D$.</td>
<td>107</td>
</tr>
<tr>
<td>5.1</td>
<td>Disconnection in two node groups by one of the critical scenarios</td>
<td>126</td>
</tr>
<tr>
<td>5.2</td>
<td>Network $G$</td>
<td>129</td>
</tr>
<tr>
<td>5.3</td>
<td>Results for ALNS with cleanUpSolution method</td>
<td>137</td>
</tr>
<tr>
<td>5.4</td>
<td>Example of a HydroGen test instance (left) and a zoom-in on part of this instance (right)</td>
<td>139</td>
</tr>
<tr>
<td>5.5</td>
<td>Plot of average log transformed objective values and cpu times for given parameter setting, grouped by instance</td>
<td>143</td>
</tr>
<tr>
<td>5.6</td>
<td>Plot of average log transformed objective values and cpu times for given parameter setting, grouped by instance (part 2)</td>
<td>144</td>
</tr>
<tr>
<td>7.1</td>
<td>The parcel distribution process from the arrival of the parcels, until the departure of the vehicles.</td>
<td>157</td>
</tr>
<tr>
<td>7.2</td>
<td>Delaunay triangulation to find natural neighbours</td>
<td>169</td>
</tr>
<tr>
<td>7.3</td>
<td>Examples of the different moves used in the variable-neighbourhood tabu search algorithm</td>
<td>170</td>
</tr>
<tr>
<td>7.4</td>
<td>Pareto-front for an instance with 200 microzones and 1 cluster, for different values of maximum deviations of the amount of work in the microzones.</td>
<td>177</td>
</tr>
</tbody>
</table>
List of Figures

7.5 The Pareto front for different initial solutions for a single instance with 200 microzones, 1 cluster and a maximum workload deviation of 15%.

7.6 Example solutions found for the instance with 150 zones, zero city centers and a deviation of workload in the zones of maximum 15% of the initial workload.

8.1 The hypervolume indicator in a two-objective case.
8.2 Linear preference and indifference area.
8.3 Residual plot for the fitted model.
8.4 Tukey HSD 95% confidence intervals for tabu list size.
8.5 Interaction plot of number of clusters and tabu list size.
8.6 Interaction plot of tabu list size and number of clusters.
8.7 Interaction plot of number of clusters and move order.
8.8 Interaction plot of number of microzones and move order.
8.9 Interaction plot of workload deviation and move order.
8.10 Box plot of move order.
List of Tables

2.1 Set of security strategies $S_i$ for edge $i = 1$ . . . . . . . 21
2.2 List of scenarios for network $G$ . . . . . . . . . . . . . . . . . . . 22
2.3 Result for an instance when estimating the objective value by restricting the critical combinations to those with a maximum of 3 failing edges. . . . . . . . . . . . . . . . . . . . 26
2.4 Heuristic parameters . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 40
2.5 Selected metaheuristic parameters . . . . . . . . . . . . . . . . . . . . . . 42
2.6 Results of the exact approach in comparison with the metaheuristic for different instances . . . . . . . . . . . . . . . . . . . . . . . . 44
2.7 Results for instances in the range 10-20 edges and 5-20 security strategies per edge . . . . . . . . . . . . . . . . . . . . . . . . . 45
2.8 Properties of the graphs shown in Figure 2.6 . . . . . . . 49
2.9 Results for the metaheuristic using critical scenarios and minimal cut sets to calculate the objective value . . . . 50
3.1 Heuristic parameters . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 75
3.2 Results of the exact approach compared with those of the metaheuristic . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 77
4.1 Set of protective measures $M_{ij}$ for edge $(i,j)$ . . . . . . . 92
List of Tables

4.2 Time to failure associated to the nodes .................. 101
4.3 List of protective safety barriers ......................... 102
4.4 Metaheuristic parameters ................................. 103
4.5 Escalation Time for all fire-paths in case of different cardinalities of domino effects .......................... 108
5.1 Critical scenarios for $G$, with and without edge $g$ ... 130
5.2 Comparison between computational times and speed-up factor (SF) for Remove I and Remove II for both the exact and ALNS approach. ............................. 131
5.3 Overview of repair neighbourhoods ....................... 136
5.4 Overview of destroy neighbourhoods ...................... 137
5.5 Legend symbols ............................................. 140
5.6 Overview of the parameter settings for the full factorial experiment ............................................. 141
5.7 Results of the exact approach compared with those of the metaheuristic ......................................... 142
7.1 Parameter values ............................................. 176
7.2 Percentage difference in hypervolume for the same problem instances with different percentages of deviation in the amount of work in the zones .................. 179
7.3 Percentage difference in hypervolume with the best found Pareto set for different start solutions and one tactical plan ...................................................... 181
7.4 Pearson’s product-moment correlation for driving distance and deviation of average worktime for instance with 200 microzones, 1 cluster and maximum workload deviation of 20% ................................. 181
List of Tables

<table>
<thead>
<tr>
<th></th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1</td>
<td>Move order</td>
<td>205</td>
</tr>
<tr>
<td>8.2</td>
<td>Top 20 ranking net flows for the GDSS procedure</td>
<td>212</td>
</tr>
</tbody>
</table>
## List of Algorithms

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Metaheuristic structure</td>
<td>30</td>
</tr>
<tr>
<td>3.1</td>
<td>Metaheuristic structure</td>
<td>66</td>
</tr>
<tr>
<td>4.1</td>
<td>Pseudo-code of the metaheuristic solution approach</td>
<td>96</td>
</tr>
<tr>
<td>5.1</td>
<td>Exact approach</td>
<td>124</td>
</tr>
<tr>
<td>5.2</td>
<td>Recursive function</td>
<td>125</td>
</tr>
<tr>
<td>5.3</td>
<td>Adaptive large neighbourhood search</td>
<td>132</td>
</tr>
<tr>
<td>5.4</td>
<td>RandomRepair function</td>
<td>134</td>
</tr>
<tr>
<td>5.5</td>
<td>GRASPRepair function</td>
<td>135</td>
</tr>
<tr>
<td>7.1</td>
<td>Diversification and storage sequence</td>
<td>171</td>
</tr>
<tr>
<td>7.2</td>
<td>Variable neighbourhood tabu search</td>
<td>172</td>
</tr>
</tbody>
</table>
Summary

Guaranteeing the delivery of a good or service is one of every service provider’s major concerns. Strategical, tactical and operational decisions should be made at different points in time. All these decisions have as a goal to increase the service provider’s service level within his/her budget. The service level can be increased in many different ways.

The presented dissertation collects the results of the author’s doctoral research, which is aimed at two facets in two different industrial sectors.

The first part of this manuscript will be focused on the work that was intended for facilitating decision makers in the utility sector with models and algorithms to help them in different stages of their network planning process. Technical problems, human mistakes or security issues can cause them to lose customers and money. To prevent their service from being disrupted, service providers try to increase the security and safety in their networks. This can be done in different stages of the service network life time and on different levels in the network.
A first problem that might be faced by a decision maker is the problem of allocating his security budget among different security strategies in an attempt to increase the security in the service network as much as possible. In a primary part, the network is modelled as a graph, with one clear source and one clear destination, of which the edges are unreliable. The assumption was made that all edges (e.g., pipes, cables) have a certain, not necessarily equal, probability of failure, which can be reduced by selecting edge-specific security strategies. In follow-up research, this model was extended in such a way that all edges and nodes (e.g., switching or connection stations, substations in an electricity network) have a certain probability of failure, which can be reduced by applying appropriate security strategies. For each of these problems, a mathematical programming model and a metaheuristic solution approach, that uses a greedy random adaptive search procedure to find an initial solution and uses tabu search hybridised with iterated local search and a variable neighbourhood descend heuristic to improve this solution, was developed. The main goal is to reduce the risk of a service failure between an origin and a destination node by selecting the right combination of security measures, given a limited security budget. The heuristic is calibrated by a statistical experiment for both problems. A tangible outcome of this research are the following papers which have been accepted for publication:

The problem of mitigating domino effects in a network by employing protective measures is investigated next. More specifically, the decision of selecting the appropriate measures and deciding where they should be placed is under assessment. Taking into account the features of an industrial area that may be affected by domino accidents, and knowing the characteristics of the safety barriers that can be installed to stall fire propagation between installations, the decision model can help practitioners in their decision-making process. The model can be effectively used to decide how to allocate a limited budget to different safety barriers. The goal is to maximize the time-to-failure of a chemical installation, using a worst case scenario approach. The model is mathematically stated and a flexible and effective solution approach, based on metaheuristics, is developed and tested on an illustrative case study representing a tank storage area of a chemical company. It was shown that a myopic optimisation approach, which does not take into account knock-on effects that are possibly triggered by an accident, can lead to a distribution of safety barriers that is not effective in mitigating the consequences of a domino accident. Moreover, the optimal allocation of safety barriers, when domino effects are considered, may depend on the so-called cardinality of the domino effects. The academic outcome of the research conducted in this chapter is the publication of the following manuscript:

A third problem that is faced by many service providers is focused on the extension of an existing network. The decision maker is faced with the problem of selecting a combination of edges that can be added to the network to increase its reliability. In this work, two approaches for distribution network design optimisation that take into account reliability at critical nodes in the network, are presented. An exact approach is compared with an adaptive large neighbourhood search solution method. Both approaches are tested on a set of realistic instances, and compared on computational time and objective value. The academic outcome of this research is presented in the subsequent paper:


The second part of the dissertation will deal with aiding the decision maker in courier companies with their operational planning based on a tactical plan. Distribution companies that serve a very large number of customers, courier companies for example, often partition the geographical region served by a depot into zones. Each zone is assigned to a single vehicle and each vehicle serves a single zone. An alternative approach is to partition the distribution region into smaller microzones that are assigned to a preferred vehicle in a so-called tactical plan. The
moment the workload in each microzone is known, the microzones can be reassigned to vehicles in such a way that the total distance travelled is minimized, the workload of the different vehicles is balanced, and as many microzones as possible are assigned to their preferred vehicle. The resulting microzone-based vehicle routing problem was modelled as a multi-objective optimization problem and a simple yet effective algorithm to solve it was developed. This algorithm and the results it obtains are discussed and analysed. The result of this academic research has been published in:


Finally, the effect of the algorithmic parameters and instance characteristics on the quality of the Pareto optimal solutions found by the aforementioned algorithm are analysed. The PROMETHEE method is used to retrieve a ranking of the possible alternatives. The academic outcome of this research is presented in the subsequent working paper:

Samenvatting

Het waarborgen van de levering van een goed of dienst is één van de belangrijkste aandachtspunten voor elke dienstverlener. Strategische, tactische en operationele beslissingen dienen op verschillende tijdstippen gemaakt te worden. Alle genomen beslissingen hebben als doel het serviceniveau van de dienstverlener binnen zijn/haar budget te verhogen. Dit kan op verschillende manieren.

Het gepresenteerde proefschrift verzamelt de resultaten van het docentaatsonderzoek van de auteur, wat zich richt op twee facetten uit verschillende industriële sectoren.

Het eerste deel van het manuscript focust zich op het werk dat gedaan is in verband met het voorzien van modellen en algoritmen die de besluitnemers in de nutssector helpen bij het nemen van beslissingen in verschillende stadia van hun netwerkplanning. Technische problemen, menselijke fouten of beveiligingsproblemen kunnen ervoor zorgen dat ze klanten en geld verliezen. Om te voorkomen dat de beschikbaarheid van hun dienst wordt verstoord, proberen dienstverleners de betrouwbaarheid en beveiliging van hun netwerk te verhogen. Dit kan tijdens
Samenvatting

verschillende stadia van het bestaan van het netwerk en op verschillende niveaus in het netwerk.

De besluitnemer wordt geconfronteerd met het toewijzen van zijn beschikbare beveiligingsbudget aan verschillende alternatieven in een poging de veiligheid in het netwerk zo veel mogelijk te verhogen. In het eerste deel van deze scriptie wordt het netwerk gemodelleerd als een graaf met één duidelijke bron en één duidelijk eindpunt, waarvoor de bogen onbetrouwbaar zijn. De veronderstelling wordt gemaakt dat alle bogen (bijvoorbeeld: leidingen, kabels) een bepaalde, niet noodzakelijk gelijke, kans op falen hebben, die kan worden verminderd door het selecteren van boog-specifieke beveiligingsstrategieën. In vervolgonderzoek, wordt het voorgaande model uitgebreid zodat alle bogen en knooppunten (bijvoorbeeld: schakel- of verbindingsstations, onderstations in een elektriciteitsnet) een kans op falen hebben, die kan worden verminderd door het selecteren van boog- of knooppunt specifieke beveiligingsstrategieën.

Voor elk van deze problemen werd een wiskundig model en een metaheuristiek, die gebruik maakt van een “greedy random adaptive search” procedure om een initiële oplossing te vinden en nadien “tabu search” gehybridiseerd met “iterated local search” en een “variable neighbourhood descend” heuristiek gebruikt om deze oplossing te verbeteren, als oplossingsmethode ontwikkeld. Het doel is het risico op falen van de dienst tussen een bron en een eindpunt te verminderen door een goede combinatie van beveiligingsmaatregelen te selecteren binnen de grenzen van een beperkt beveiligingsbudget. De heuristiek werd voor beide problemen gekalibreerd met behulp van een statistisch experi-
Samenvatting

Een tastbaar resultaat van dit onderzoek zijn de volgende manuscripten die zijn geaccepteerd voor publicatie:


Het probleem waarbij domino-effecten horen beperkt te worden door het toepassen van veiligheidsmaatregelen is vervolgens onder de loep genomen. Meer specifiek wordt het uitzoeken van geschikte maatregelen en selecteren van de correcte plaats om deze toe te passen onderzocht. Op basis van de eigenschappen van een industrieel gebied dat kan worden getroffen door domino-ongevallen en de kennis van de karakteristieken van de veiligheidsmaatregelen die kunnen worden geïnstalleerd om voortgang tussen installaties te vertragen, kan het beslissingsmodel de besluitnemers helpen in hun besluitvorming. Het model kan doeltreffend ingezet worden om te beslissen hoe een beperkt budget toegewezen dient te worden aan veiligheidsmaatregelen. De besluitnemer heeft als doel de time-to-failure van een chemische installatie te maximaliseren. Het model werd wiskundig verklaard en een flexibele en efficiënte oplossingsmethode, gebaseerd op metaheuristieken, werd ontwikkeld en getest op een illustratieve case study die de tanks in een opslagruimte van een chemisch bedrijf voorstelt.

xxv
Samenvatting

Er wordt aangetoond dat een kortzichtige optimalisatiemethode, waarbij geen rekening wordt gehouden met domino-effecten die mogelijk veroorzaakt worden door een ongeval, kan leiden tot een verdeling van de veiligheidsmaatregelen die niet efficiënt de gevolgen van een domino-ongeval vermindert. Bovendien kan de optimale verdeling van veiligheidsmaatregelen, wanneer domino-effecten worden beschouwd, afhangen van de zogenaamde kardinaliteit van deze kettingreactie. Het onderzoek leverde volgende publicatie op:


Een derde probleem waar vele dienstverleners mee geconfronteerd worden, is gerelateerd aan het uitbreiden van een bestaand netwerk. De besluitnemer dient een combinatie van bogen te selecteren die aan het netwerk kunnen toegevoegd worden, teneinde de betrouwbaarheid zo veel mogelijk te verhogen. In dit werk, worden twee oplossingsmethoden voorgesteld voor het optimaliseren van het distributienetwerk tijdens de ontwerpfase, rekening houdende met de betrouwbaarheid van de levering van een dienst aan bepaalde kritieke knooppunten in het netwerk. Een exacte aanpak wordt vergeleken met een “adaptive large neighbourhood search” oplossingsmethode. Beide benaderingen worden getest op een reeks van realistische problemen, en vergeleken op rekentijd en doelfunctiewaarde. Het academische resultaat van dit onderzoek werd gepresenteerd in het volgende artikel:
Samenvatting


Het tweede deel van het proefschrift richt zich op het helpen van de besluitnemer in koeriersbedrijven bij hun operationele planning op basis van een tactische plan. Distributiebedrijven die een zeer groot aantal klanten bedienen, koeriersdiensten bijvoorbeeld, verdelen de geografische regio, die door een depot bediend wordt, vaak op in zones. Elke zone wordt toegewezen aan één enkel voertuig en aan elk voertuig wordt één enkele zone toegewezen. Een alternatieve benadering, die hier onder de loep wordt genomen, is er een waarbij de distributieregio wordt opgedeeld in kleinere micro-zones die zijn toegewezen aan één geprefereerd voertuig in een zogenaamd tactisch plan. Zodra de werkdruk in elke micro-zone bekend is, kunnen de micro-zones worden toegewezen aan voertuigen, zodanig dat de totale afgelegde afstand wordt geminimaliseerd, de werklast van de verschillende voertuigen evenwichtig is en zoveel mogelijk micro-zones worden toegewezen aan hun geprefereerde voertuig. Het resulterende op micro-zones gebaseerde rittenplanningsprobleem werd gemodelleerd als een optimaliseringsprobleem met meerdere doelstellingen en een eenvoudig, maar zeer efficiënt algoritme om het op te lossen werd ontwikkeld. De resultaten van het onderzoek worden gepresenteerd in het volgende artikel:

Ten slotte, werd het effect van de parameters van het algoritme en de instances op de kwaliteit van de pareto-optimale oplossingen die gevonden werden door het voorgaande algoritme geanalyseerd. De Promethee-methode wordt gebruikt om een rangschikking te maken van de mogelijke alternatieven. Het academische resultaat van dit onderzoek werd gepresenteerd in het volgende artikel:

Part I

NETWORK SECURITY OPTIMISATION
Introduction
In modern day society, utility networks such as electricity, water, gas, and communication networks are taken for granted. People expect that they function at all times, and are capable of handling all demand that is placed on them. Guaranteeing the delivery of a good or service is one of every service provider’s major concerns. Technical problems, human mistakes or security issues can cause them to lose customers and money. There is a real risk of these types of failure in all sorts of networks. Those failures might interrupt the service/connection between origin nodes (i.e., the points from which the service or the product is sent to the customer through the network) and destination nodes (i.e., the customer or points to which the product or service is delivered through the network).

Utility networks are critical infrastructures that deserve increased attention to limit the consequences of failures since they may result into serious breakdowns and can cascade into dependent systems (Kröger, 2008). For example, the blackout in India in 2012 left more than 600 million people without access to energy. This emphasizes the need for adequate protective measures against network breakdowns (Barker et al., 2013; Romero, 2012).

Network breakdowns can have safety-related (i.e., unintentional) causes such as natural phenomena (e.g., earthquakes, storms), human errors, or mechanical defects such as in pumps and valves, caused by the regular wear and tear. In addition, network breakdowns can be due to security-related causes such as intentional terrorist attacks and/or malicious sabotage. The reader is referred to Reniers et al. (2008) for a further clarification of terms safety and security. For simplicity, in Chapter 2
and Chapter 3, mainly network security will be focussed on, however, the solution methods proposed in those chapters are also applicable to mitigate safety related causes.

To prevent their service from being disrupted, service providers try to increase the security and safety in their networks. *This can be done in different stages of the service network life time, or on different levels in the network.*

During the planning stage, the decision maker could wonder: “Where should I add connections to increase my capacity? How can I prevent a disruption of my service to a certain area?” The answers to these questions are directly related to the cost of each alternative.

The focus of this part of the manuscript, however, is on the increase of safety and security in an *existing* network. Here too the decision maker can wonder about various things: “Is it better to add a security measure on this edge or on that one? Should I add redundant edges to increase reliability in my network? Where should I add these redundancies? How can I decrease and delay cascading effects of an incident in my network?” Again the answer is related to the cost of each alternative. For the decision maker, however, these are relevant questions, and they are difficult to answer due to the massive number of possible alternatives.

The problems discussed in this part of the manuscript are all combinatorial optimisation problems. Several models and solution methods to guide the decision maker in the process of allocating his budget in an intelligent manner are proposed.
In Chapter 2 and 3, the decision maker is facing a problem of allocating his budget among different security strategies in an attempt to increase the reliability in the service network as much as possible. The former chapter defines the security network with one clear source and one clear destination. In the latter, every node can be regarded as a source or a destination. A variable neighbourhood tabu search (VNTS) method is created and tailored to the specifications of each of these problems.

Chapter 4 focuses on the problem of mitigating domino effects in a network by employing protective measures. More specifically, the decision of selecting the appropriate measures and deciding where it should be placed is studied.

In Chapter 5 the extension of an existing network is investigated. The decision maker is faced with the problem of selecting a combination of edges that can be added to the network to increase its reliability. A mathematical model is presented and an adaptive large neighbourhood search (ALNS) metaheuristic is proposed to solve it.
Models and algorithms for the optimisation of security strategies in networks with a single source and destination node

This chapter is based on the manuscript “Janssens, J., Talarico, L., and Sörensen, K. (2016c). A hybridised variable neighbourhood tabu search heuristic to increase security in a utility network. Reliability Engineering & System Safety, 145:221 – 230”
2.1 Introduction

After 9/11, the protection of utility networks against intentional attacks has received great attention among network providers. Terrorist attacks on utility networks are not rare and can cause huge losses in a nation’s economy (START, 2014).

Network providers and managers can reduce the risk of a network breakdown, due to failures in one or several network edges, by applying preventive measures to reduce network vulnerabilities. These preventive measures can be, amongst others, fences, camera’s or security patrols to increase network security and redundancy or stronger materials and newer technologies for an increase in network safety.

The security budget that can be spent on these security measures, however, is generally limited. The current economic situation has increased the pressure on limiting many budgets and investments in security even further. In this chapter, a combination of security measures applied to an edge is called a security strategy for that edge.

This chapter proposes and analyses a method to efficiently allocate a budget among a set of edge-specific security strategies in order to minimise the risk of a disconnection between one given origin node and one given destination node. In the remainder of this chapter, the utility network is considered down whenever the origin and the destination node are disconnected from each other. More specifically, this means that no path exists between the origin and destination nodes. Furthermore, only edges can be subject to failure.
Since the budget is limited and the security strategies can only be applied locally, i.e., on specific links in the network, the security strategies should be chosen in such a way that the reduction of the risk of the network service being down is as large as possible while keeping the total cost of the security strategies within the budget.

Once realistic cases with a large number of edges and different security strategies (from 5 up to 20) are considered, the problem becomes computationally infeasible to solve in a reasonable amount of time with exact algorithms. Therefore, the use of a metaheuristic approach to support this decision problem is explored.

The remainder of this chapter is organized as follows. In Section 2.2, we give a brief overview of the state of the art. Section 2.3 clarifies the problem of selecting the best strategies to increase the security of the whole network. It is described and modelled as an optimisation problem. An illustrative example is also provided. In Section 2.4, we present a metaheuristic to solve the network security problem. Section 2.5 presents the instance generator, reports the parameter tuning phase and the results of the computational experiments. In Section 2.6, an alternative model and solution approach is presented. Section 2.7 concludes the chapter.

2.2 Literature review

To guarantee the robustness of networks, a lot of research has been conducted in the field of survivable networks (see Steiglitz et al., 1969; Kerivin and Mahjoub, 2005). Most work in this field attempts to improve
network robustness by means of redundant paths between a single set of nodes, without taking probabilistic information into account. They do this by adding edges to the network. The approach proposed in this chapter takes a fixed network into account, and mitigates the risk of failure by adding security measures on existing edges in the network. Although significant research has been done to improve best practices in the field of security, a minority of the work has addressed the relationship between risk-related variables and cost-effective network security decisions that impact the objective. The major part of it deals with the problem of designing more robust and reliable utility networks in a preliminary stage, in which a network designer tries to decrease the probability of service failures by increasing the number of redundant links in the network. Only a small section, of which a selection are discussed below, has addressed the problem of supporting decision makers to protect an existing utility network from external attacks by investing in effective security measures, subject to a budget constraint and technical limitations.

As mentioned, this chapter proposes a decision model to support utility managers in the selection of cost-effective network protection investments. A similar approach has been proposed by Rees et al. (2011) for the field of IT. In their work, security countermeasures are used to guarantee the confidentiality, availability, and integrity of data in computer systems that might be subject to cyber-attacks. A decision support system, based on a genetic algorithm, is proposed to select the “best” combination of countermeasures that respects the user’s preferred trade-off between the cost of the selected security measures and the resulting risk exposure of the computer system. As they are working in a single
system. Their objective value is defined as the sum of the product of expected events of type i and the cost in dollars for event i, for all types of events. As they are working in a single system, spatial information is of no importance.

The problem of selecting the right security measures given a limited budget is clearly not an easy task. Most security planning models in the literature are qualitative and only few of them rely on quantitative approaches. In case of a pipeline network, the security risk assessment procedure defined by Reniers and Dullaert (2012) may be used. After a careful pipeline security risk assessment, the user is in possession of pipeline segment risk data as well as pipeline route risk data. Assuming that the security risk analyst determines a set of available security measures and defence strategies for application to the different pipeline segments and/or for the pipeline routes, a selection of the most effective security measures with respect to the available budget (either for a single pipeline segment or for a pipeline route) can be calculated. If the cost of the security measures are known in advance, a mathematical approach can be used to solve the problem of optimal allocation of security resources by solving a knapsack problem with additional constraints. Reniers et al. (2012) explain how this well-known problem in the field of Operations Research can be used in security optimisation problems. A practical application to secure an illustrative pipeline infrastructure used to transport oil is described in Talarico et al. (2015). In the works of Reniers and Dullaert (2012), Reniers et al. (2012) and Talarico et al. (2015), methods for the assessment of the risk of failure are proposed. The design and implementation details of the solution method to reduce this risk by means of the application of security meas-
ures, however, is left (partially) unresolved.

In Bistarelli et al. (2007) a method for the identification of the assets, threats and vulnerabilities of ICT systems is introduced. Furthermore, a qualitative approach for the selection of security measures to protect an IT infrastructure from external attacks is discussed. In particular, two security models based on defence trees (an extension of attack trees) and preferences over security measures are proposed. Here too, there is no spatial information used, as they deal with one system. Furthermore, no budget constraint is present in their model.

In Viduto et al. (2012) the security of a telecommunication network is analysed from a quantitative point of view. Knowledge of potential risks enables organisations to take decisions on which security measures should be implemented before any potential threat can successfully exploit system vulnerabilities. A security measure selection problem is presented in which both cost and effectiveness of an implemented set of security measures are addressed. A Multi-Objective Tabu Search (MOTS) algorithm is developed to construct a set of non-dominated solutions which can satisfy organisational security needs in a cost-effective manner. Again, no spatial information is necessary, as the solution method is designed for attacks on a single system.

In Sawik (2013), a similar security measure selection problem for an IT infrastructure is formulated as a single- or bi-objective mixed integer programming problem. Given a set of potential threats and a set of available measures, the decision maker needs to determine which ones to implement, under a limited budget, to minimize potential losses from successful cyber-attacks and mitigate the impact of disruptions caused
by IT security incidents. Their approach is very similar to the one proposed in this chapter. The main difference is that the security measures, in their approach, are applied on the whole system, not on a single edge. This means that the application of a countermeasure impacts multiple threats. The fact that each edge can have a different set of available security measures, makes the problem and solution method proposed in this chapter more generic.

The prevention of heavy losses caused by cyber-attacks and other information system failures in an IT network is usually associated with continuous investment in different security measures. In Bojanc and Jerman-Blazič (2008), several approaches enabling the assessment of the necessary investments in security technology are addressed from an economical point of view. Their paper introduces methods for the identification of risks in ICT systems and proposes a procedure that enables the selection of the optimal level of investments in security measures.

Once security risks have been identified, the potential loss associated with their occurrence, as well as their probability of occurrence must be determined. Probability theory is used extensively in reliability theory and in reliability studies of systems. For an overview, I refer the reader to Bazovsky (2004); DES JSC TLS POL REL (2011); Romeu (2004). Determining both the probability of occurrence and potential impact of each risk is done in a process called risk assessment. Performing a risk assessment phase allows to take decisions regarding the necessary investment in security controls and systems. In this chapter, a preliminary risk assessment phase is assumed to have been previously
conducted by experts, a second assumption is that they defined the probability for each edge to fail together with the costs and benefits of each available security measure for that edge. A methodology to conduct risk assessment for safety-related accidents has been proposed in Antonioni et al. (2009). In Reniers et al. (2014) and Reniers et al. (2013), security-related risk and threat assessment within chemical plants is studied.

In Agarwal et al. (2013), the focus is on probabilistic attacks and on multiple simultaneous attacks on telecommunication networks. The probabilities associated to these events are dependent on geographical locations. Vulnerable points within the network are identified by applying geometric tools. Using this approach, it is possible to identify locations which require additional protection efforts.

The work of Reniers and Dullaert (2012) gives a nice overview of risk assessment for security-related attacks on pipeline networks. For each pipeline segment the risk exposure is computed considering the features of that segment and the hypothetical consequences of an accident scenario. For many of the aforementioned publications which only concern themselves with the risk assessment, the approach proposed in this chapter represents a next step in the process. It defines a single-objective problem and proposes a quantitative method to select appropriate security measures.

In Reniers et al. (2012) an approach aimed at formulating recommendations to more effectively protect a chemical cluster against existing systemic risks and decrease interdependent risks within chemical industrial areas, is proposed. A multi-attribute method is developed mod-
elling chemical industrial areas as interconnected and complex networks and using a holistic optimization approach considering inter-organisational and inter-cluster objectives. The goal of the model is to use a quantitative approach to map systemic risks in chemical industrial areas. Using the outcome of the model risk, experts and decision makers can make a quantitative risk assessment to objectively inform themselves about possible prevention measures to lower the risk exposure. However, after having analysed the network topology, the selection of the most appropriate safety measures is left to the decision maker.

The approach in this chapter uses an objective function which relies on the minimization of the risk of the network to be not accessible between a source node and a destination node. Moreover, in this work, since a list of security measures is defined for each edge of the network, the model incorporates not only decisions taken at the level of the network, as done in Reniers and Dullaert (2012), Reniers et al. (2012) and Sawik (2013), but it depends on the choices made at the level of single network edges. A mathematical model and efficient metaheuristic solution approach, for the problem introduced in this chapter, are proposed. The solution method is rigorously tested on different test instances, and the results are compared to those of an exact approach.

### 2.3 Problem description

A utility network can be represented by a graph $\mathcal{G} = \{\mathcal{N}, \mathcal{E}\}$, where $\mathcal{N}$ represents a set of nodes and $\mathcal{E}$ a set of edges, connecting the nodes.
All edges $i \in \mathcal{E}$ have a probability of failing, denoted as $p_i$. A set of security strategies $\mathcal{S}_i$, is defined for each edge $i \in \mathcal{E}$.

For each security strategy $j \in \mathcal{S}_i$ of edge $i$, a cost $c_{ij}$ is given, as well as the probability of failure of edge $i$ given that security strategy $j$ has been chosen, $p_{ij}$. Exactly one security strategy per edge can be selected. Each edge also has a default “do-nothing” strategy $j = 0$ with cost $c_{i0} = 0$.

The model proposed in this chapter makes the assumption that only edges are vulnerable to failure and that nodes are not.

An origin node $o$ and a destination node $d$ in the network, are given. The quality of a solution (e.g., a selection of a security strategy for each edge) is defined as the probability that no path exists between node $o$ and node $d$. This would make it impossible for a service or good from node $o$ to reach node $d$ (e.g., it would be impossible to make a phone call from node $o$ to node $d$). The goal is to minimise this probability.

Since the decision problem is introduced for the first time, in this chapter, the problem is simplified by making the assumption that only one supplier and one customer exist in the network. In a realistic network, the network operator is not only dealing with the question if there is a connection to a customer, but also with the issue of having enough capacity to deliver his product or service in a reliable way. In this chapter, since we are dealing with only one customer, an abstraction is made of this last problem. The assumption is made that the load placed on the network by the customer is considerably lower than the load that each network section (edge) can handle. This reduces the
Chapter 2 Single source and destination security strategy optimisation

problem to a much simpler “path or no path” problem. In Chapter 3, this model is extended to one where multiple source and destination nodes are considered. A further extension, which is not considered in this manuscript, could be to consider supplier capacity, customer demand and the importance of either of them.

To calculate the probability that no path exists between a given origin and a given destination node, probability theory is used.

2.3.1 Illustrative Example

A security strategy can be one or a combination of individual security measures (see e.g., Table 2.1, where an example set of security strategies $S_i$ for edge $i = 1$ is defined with for each of them the respective cost $c_{ij}$ and probability $p_{ij}$). A combination of security measures can have a different effectiveness than the sum of the impact of the individual security measures due to interaction effects. In some cases, combinations of single security measures might not be available due to their incompatibility. The probabilities defined for each of the example strategies, in Table 2.1, are merely there for illustrative purposes. Realistic values of the impact of security measures on the safety or security of a network are, to the best of our knowledge, not available or researched yet. In the remainder of this work, we make the assumption that we have data available, provided by a risk assessment, executed by an expert, for each of the instances used.

To clarify the computation used to determine the total risk that no path exists between nodes $o$ and $d$, an example graph is shown in Figure 2.1.
A scenario is defined as a single edge or a combination of edges that fail at a certain point in time. Given the topology of the example network, there are eight possible scenarios. Three of them are critical, which means they that remove every path between nodes $o$ and $d$. The cases in which no path exists in the network between nodes $o$ and $d$, are the critical scenarios 4, 5 and 7. To find these scenarios, all possible combinations of failing edges are considered and tested. Table 2.2 contains all scenarios $l$. 

---

**Table 2.1: Set of security strategies $S_i$ for edge $i = 1$**

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Security measures</th>
<th>Cost</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0.70</td>
</tr>
<tr>
<td>1</td>
<td>Fences</td>
<td>100</td>
<td>0.50</td>
</tr>
<tr>
<td>2</td>
<td>Camera type A</td>
<td>150</td>
<td>0.45</td>
</tr>
<tr>
<td>3</td>
<td>Camera type B</td>
<td>200</td>
<td>0.40</td>
</tr>
<tr>
<td>4</td>
<td>Fences &amp; Camera type A</td>
<td>250</td>
<td>0.32</td>
</tr>
<tr>
<td>5</td>
<td>Fences &amp; Camera type B</td>
<td>300</td>
<td>0.25</td>
</tr>
</tbody>
</table>

---

---

**Figure 2.1:** Utility network $\mathcal{G}$ with a source $o$ and a destination $d$. 

---

21
All scenarios $l$ are defined as a combination of edges that fail, contained in set $E^F_l$, and edges that do not fail, contained in set $E^N_l$. A critical scenario is a scenario where the edges that do fail, if they fail all at once, will disable every path between node $o$ and node $d$. The set of critical scenarios is defined as $C$. For all scenarios the following property holds: $E^F_l \cup E^N_l = E$. The cardinality of set $C$ depends on the topology of the network $G$ and the position of nodes $o$ and $d$ within the network. Edge $p_i$ in Table 2.2 corresponds to the edge with probability $p_i$ in Figure 2.1. In the table, the value in column $p_i$ is 1 below $E^F_l$, if edge $p_i$ is failing in scenario $l$, and 0 otherwise. For $E^N_l$ the same logic holds true. A failure, also called an event, is defined as a malfunctioning of a part or section of a network, with as a results the stop of the flow of a good or service through that part or in that section. In the figure, the probability of a failure happening is reported near each edge it is associated to.

<table>
<thead>
<tr>
<th>$l$</th>
<th>$p_1$</th>
<th>$p_2$</th>
<th>$p_3$</th>
<th>$p_1$</th>
<th>$p_2$</th>
<th>$p_3$</th>
<th>$R_l$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.144</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.336</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.096</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0.036</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.224</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.084</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.024</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.056</td>
</tr>
</tbody>
</table>

For example, the first line in Table 2.2 shows a scenario where no edges

22
fail, hence all paths between node $o$ and node $d$ exist. The fourth line, however, shows a scenario where both edges 1 and 2 fail. Therefore, no path exists from node $o$ to node $d$, even if edge 3 does not fail. The probability $R_1$ associated to the occurrence of this scenario, is equal to $p_1 \cdot p_2 \cdot (1 - p_3)$ that is $0.4 \cdot 0.7 \cdot (1 - 0.2) = 0.224$. The same reasoning applies to the remaining scenarios reported in Table 2.2, which contains all possible scenarios for this graph.

As we are dealing with every possible scenario and the scenarios are independent events, it is possible to compute the total probability of a disconnection between node $o$ and node $d$ as the sum of the probabilities of occurrence of all critical scenarios, defined as $\sum_{l \in C} R_l$. In this example, scenarios 4, 5 and 7, for which the sum is equal to 0.364. Each time a security strategy is changed or updated, the probability for each scenario should be recalculated. And the objective function, the total probability for a disconnection between node $o$ and node $d$, also changes.

As expected, the computation time needed to update the objective function every time a move is applied, grows exponentially if the number of edges increases (see Figure 2.2(a)). This is due to the number of critical scenarios, which grows significantly depending on the number of edges and on the topology of the network. As shown in Figure 2.2(b), the relationship between the number of critical scenarios in $C$ and the computational time is approximately linear.

Despite the fact that the update of the objective function is expensive in terms of computational time, an estimation of the objective value proves to be difficult. A naive approach to estimate the objective value
Chapter 2 Single source and destination security strategy optimisation

Figure 2.2: Relationship between the number of edges (a) and critical scenarios (b) and CPU time needed to update the objective function after a move is executed.

could be the restriction of the critical scenarios to scenarios with a limited number of edges that fail, as scenarios with a higher number of failing edges have a low probability to occur. This approach, however, can result in unexpected behaviour. Depending on the topology of the network and the values assigned to edges and security strategies, it is possible that the objective value for the network, where all security strategies are the “do-nothing” strategy, is lower than the objective value of the network where one security strategy, for one edge, is selected. This is counter-intuitive as the selection of a security strategy should always lower the objective value in comparison to the objective value where all security strategies are the “do-nothing” strategy. A small example is given in Figure 2.3. All edges have a probability of failure equal to 1%. Edge $E$ is allocated a security strategy, which reduces its probability of failure to 0.7%. Table 2.3 shows all the critical combinations where at most 3 edges fail at the same time. The first
column indicates which edges are failing. The second column contains the probability of occurrence for that critical combination when the security strategy is not applied. The third column contains the probability of manifestation when the security strategy is applied on edge $E$. The bottom line of the table contains the objective value. As the problem is a minimisation problem, this table shows that when the security strategy is applied to edge $E$, the objective value worsens.

The graph presented in Figure 2.3 results in 512 scenarios, ranging from no edges failing to all edges failing, from which 350 are critical. If we restrict the list to critical scenarios with a maximum of 3 edges failing, we reduce this number to 32 scenarios, shown in Table 2.3. Looking at the edges in all the scenarios, it becomes clear that if there is no limit in the number of edges in a critical scenario, 53 percent of the critical scenarios contain edge $E$ as a failing edge, while with the limit, only 19 percent incorporates this edge. As shown in Section 2.3.2, the objective value is a summation of the probability of all critical scenarios. The probability is reduced when the edge is present in the scenario, but slightly increases when the edge is not in the scenario (it increases the risk of not failing, which is the second part of Equation (2.4) in Section 2.3.2). Because with the restriction to critical scenarios with a maximum of 3 failing edges only 19 percent will decrease in probabil-
Table 2.3: Result for an instance when estimating the objective value by restricting the critical combinations to those with a maximum of 3 failing edges.

<table>
<thead>
<tr>
<th>Failing edges</th>
<th>no security measure (in $10^{-5}$)</th>
<th>security measure (in $10^{-5}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A C</td>
<td>9.32</td>
<td>9.35</td>
</tr>
<tr>
<td>A D</td>
<td>9.32</td>
<td>9.35</td>
</tr>
<tr>
<td>B C</td>
<td>9.32</td>
<td>9.35</td>
</tr>
<tr>
<td>B D</td>
<td>9.32</td>
<td>9.35</td>
</tr>
<tr>
<td>A B C</td>
<td>0.09</td>
<td>0.09</td>
</tr>
<tr>
<td>A B D</td>
<td>0.09</td>
<td>0.09</td>
</tr>
<tr>
<td>A C D</td>
<td>0.09</td>
<td>0.09</td>
</tr>
<tr>
<td>A C E</td>
<td>0.09</td>
<td>0.07</td>
</tr>
<tr>
<td>A C F</td>
<td>0.09</td>
<td>0.09</td>
</tr>
<tr>
<td>A C G</td>
<td>0.09</td>
<td>0.09</td>
</tr>
<tr>
<td>A C H</td>
<td>0.09</td>
<td>0.09</td>
</tr>
<tr>
<td>A C I</td>
<td>0.09</td>
<td>0.09</td>
</tr>
<tr>
<td>A D E</td>
<td>0.09</td>
<td>0.07</td>
</tr>
<tr>
<td>A D F</td>
<td>0.09</td>
<td>0.09</td>
</tr>
<tr>
<td>A D G</td>
<td>0.09</td>
<td>0.09</td>
</tr>
<tr>
<td>A D H</td>
<td>0.09</td>
<td>0.09</td>
</tr>
<tr>
<td>A D I</td>
<td>0.09</td>
<td>0.09</td>
</tr>
<tr>
<td>B C D</td>
<td>0.09</td>
<td>0.09</td>
</tr>
<tr>
<td>B C E</td>
<td>0.09</td>
<td>0.07</td>
</tr>
<tr>
<td>B C F</td>
<td>0.09</td>
<td>0.09</td>
</tr>
<tr>
<td>B C G</td>
<td>0.09</td>
<td>0.09</td>
</tr>
<tr>
<td>B C H</td>
<td>0.09</td>
<td>0.09</td>
</tr>
<tr>
<td>B C I</td>
<td>0.09</td>
<td>0.09</td>
</tr>
<tr>
<td>B D E</td>
<td>0.09</td>
<td>0.07</td>
</tr>
<tr>
<td>B D F</td>
<td>0.09</td>
<td>0.09</td>
</tr>
<tr>
<td>B D G</td>
<td>0.09</td>
<td>0.09</td>
</tr>
<tr>
<td>B D H</td>
<td>0.09</td>
<td>0.09</td>
</tr>
<tr>
<td>B D I</td>
<td>0.09</td>
<td>0.09</td>
</tr>
<tr>
<td>E F I</td>
<td>0.09</td>
<td>0.09</td>
</tr>
<tr>
<td>E H I</td>
<td>0.09</td>
<td>0.09</td>
</tr>
<tr>
<td>F G I</td>
<td>0.09</td>
<td>0.09</td>
</tr>
<tr>
<td>G H I</td>
<td>0.09</td>
<td>0.09</td>
</tr>
</tbody>
</table>

Objective value | 39.92 | 39.93
ity, and the rest will have a slight increase, depending on the problem instance and the values assigned to the probabilities for each edge and the reduction for the applied security strategy, this might result in the increase of the objective value, while a reduction is expected.

### 2.3.2 Mathematical model

In order to mathematically state the decision problem associated to the selection of the best set of security strategies to increase the overall network security, the risk of no path being available between node $o$ and destination $d$ has to be specified. For this reason, a set $C$, the set of critical scenarios, is defined.

\[
\text{min} \quad \sum_{l \in C} R_l \tag{2.1}
\]

s.t.

\[
\sum_{i \in \mathcal{E}} \sum_{j \in \mathcal{S}_i} c_{ij} \cdot x_{ij} \leq B \tag{2.2}
\]

\[
p_i = \sum_{j \in \mathcal{S}_i} p_{ij} \cdot x_{ij} \quad \forall i \in \mathcal{E} \tag{2.3}
\]

\[
R_l = \prod_{i \in \mathcal{E}_l^F} p_i \cdot \prod_{k \in \mathcal{E}_l^N} (1 - p_k) \quad \forall l \in C \tag{2.4}
\]

\[
\sum_{j \in \mathcal{S}_i} x_{ij} = 1 \quad \forall i \in \mathcal{E} \tag{2.5}
\]

\[
x_{ij} \in \{0, 1\} \quad \forall i \in \mathcal{E}, \forall j \in \mathcal{S}_i \tag{2.6}
\]
Let $B$ represent the available security budget and $x_{ij}$ a binary variable, that takes value 1 when the security strategy $j$ on edge $i$ is applied, and 0 otherwise.

The objective function (2.1) minimizes the total risk that no path exists between nodes $o$ and $d$. The total network risk is given by the sum of risks associated to single scenarios happening, as these scenarios are statistically independent events. Constraint (2.2) ensures that the total cost associated to the selected security strategies does not exceed the predefined security budget $B$. Equation (2.3) is used to define the probability $p_i$ associated to a failure of edge $i$, given a selected security strategy for that edge. Equation (2.4) defines the risk that a scenario $l$, which disconnects all paths in the network between nodes $o$ and $d$, takes place, for all $l$ in the set of critical scenarios. Equation (2.5) forces the decision process to select at maximum one security strategy to protect each edge $i$. When $x_{i0}$ is set equal to 1, no security strategy is applied for edge $i$. Finally, constraint (2.6) represents the domain of the decision variable, which ensures that no partial security strategies are allowed.

### 2.4 Solution approach

The decision problem of selecting appropriate security strategies given a budget constraint, in order to reduce the risk of disconnection between node $o$ and node $d$, belongs to the more general category of knapsack problems.
2.4 Solution approach

More specifically, since the objective function is not linear, it belongs to the class of non-linear knapsack problems, also known as the non-linear resource allocation problem, which belongs to the category of combinatorial optimization problems (Bretthauer and Shetty, 2002).

The metaheuristic that has been developed in this chapter is shown in Algorithm 2.1. Our variable neighbourhood tabu search (VNTS) metaheuristic is composed of three consecutive steps: (1) a greedy randomized adaptive search procedure (GRASP) is used during the constructive phase (see Section 2.4.1) to generate an initial solution; (2) a variable neighbourhood descent (VND) is used during the improvement phase (see Section 2.4.2) to improve the solution generated by the GRASP and; (3) two perturbation heuristics are used during the diversification stage (see Section 2.4.3) to escape from local optima. In addition, a tabu list is used during the whole execution of the heuristic to avoid the exploration of solutions that have been analysed in previous iterations.

The first step of this iterative solution approach consists of running a GRASP constructive heuristic.

After the GRASP procedure is finished, a local search is used to improve the current solution by using a VND heuristic. This local search is executed until the algorithm finds no more improvement. Once this is the case, a perturbation is applied to escape the local optimum, and the algorithm continues with a local search from this perturbed solution. If after a predefined number of perturbations no better solution can be found, the algorithm is restarted from a new solution generated by the GRASP heuristic. After a maximum number of iterations, which is established in advance, the algorithm is terminated, and the best found solution is returned to the decision maker.
Algorithm 2.1 Metaheuristic structure

1: Initialise both Problem and Heuristic parameters
2: \( x \leftarrow \text{GRASP}() \)
3: \( x^* \leftarrow x, f(x^*) \leftarrow f(x) \)
4: \( \text{max-iter-no-improvement} \leftarrow 0 \)
5: \( \text{iterations} \leftarrow 0 \)
6: while (\text{iterations} < \text{max-iter}) do
7: \( x \leftarrow \text{VND}(x) \)
8: if \( (f(x) < f(x^*)) \) then
9: \( x^* \leftarrow x, f(x^*) \leftarrow f(x) \)
10: \( \text{max-iter-no-improvement} \leftarrow 0 \)
11: else
12: \( \text{max-iter-no-improvement} \leftarrow \text{max-iter-no-improvement} + 1 \)
13: end if
14: if \((\text{max-iter-no-improvement} < \text{MaxPerturbations})\) then
15: \( x \leftarrow \text{Perturbation}(x) \)
16: else
17: \( x \leftarrow \text{GRASP}() \)
18: end if
19: \( \text{iterations} \leftarrow \text{iterations} + 1 \)
20: end while
21: return \( x^* \)

2.4.1 Constructive phase

The solution approach, proposed here, uses a greedy randomized adaptive search procedure (GRASP) to generate an initial solution \( x \).

GRASP (Feo and Resende, 1989) is a well-known constructive heuristic, that, starting from an empty solution, generates a complete solution by upgrading one strategy at a time until either the security budget has
been depleted or no more security strategy upgrades are available.

The criticality of an edge is defined as the value measuring the importance of that edge for the security of the network. The higher the criticality of an edge, the greater the impact of that edge on the risk reduction and thus on the improvement of the objective function. In this algorithm, the number of times that an edge occurs in a critical scenario multiplied by the remaining probability of failure for that edge, after the application of a security strategy, is used as an estimate of criticality.

At each iteration of the GRASP heuristic, the strategy to be upgraded in the current solution is randomly selected from a restricted candidate list (RCL). The RCL contains the first $\alpha$ critical edges ordered by decreasing value of criticality, for which: (a) no security strategy has already been included in the current solution and (b) at least one security strategy, not contained in the tabu list and whose cost is lower than the remaining budget, is available.

The size of the RCL, $\alpha$, is a parameter that controls the balance between greediness and randomness. If $\alpha$ is large, the selection is relatively random, while if $\alpha = 1$ the greedy construction is completely deterministic.

In this way, at each iteration of the GRASP heuristic the RCL contain the first $\alpha$ edges that are more likely to have a greater impact on the total risk considering the overall network. An edge $i$ is randomly chosen from this RCL and an available strategy $j$ whose cost is lower
than the remaining budget is selected for that edge. Two possible selection mechanisms can be employed to select a strategy at each iteration of the heuristic:

- **Best improvement** provides at each iteration the combination \((i, j)\) which yields the largest reduction in the objective function given the \(\alpha\) considered edges in the RCL and their available security strategies; This is more demanding in terms of computational time.

- **First improvement**, which guarantees a fast selection of a feasible combination \((i, j)\). From the \(\alpha\) edges, one edge is chosen in a random manner and for this edge a security strategy is selected at random. This selected combination \((i, j)\) is then checked for feasibility, which means it is not used in the current solution, not contained in the tabu list and is less costly than the remaining budget. If the selected combination is not feasible, a new combination is selected in a random fashion, until a feasible combination is found. This combination provides a reduction of the risk for the overall network, but is not necessarily the highest possible reduction, as the combination is randomly selected.

During the constructive method, a tabu list is used in order to exclude the selection of security strategies that have already been explored in previous iterations of the algorithm. After the selection process, the current solution, remaining budget and RCL are all updated. The selected edge \(i\) is removed from the RCL, as well as all the edges for which no feasible strategy is available any longer due to the reduced remaining budget. Then, the RCL is reordered based on the new vulnerabil-
ity values of the edges. The GRASP heuristic’s selection mechanism is repeated until either the remaining budget is consumed or the list of available candidates is empty.

### 2.4.2 Improvement phase

The improvement of the solution, produced by the GRASP heuristic, is performed by a variable neighbourhood descent (VND) heuristic. VND is a deterministic variant of the well-known variable neighbourhood search (VNS) metaheuristic (Hansen and Mladenović, 2001). In general, VNS algorithms use a sequence of nested neighbourhood, \( N_1, \ldots, N_{k_{\text{max}}} \) with an increasing size, i.e., \( |N_k| < |N_{k+1}| \) and a perturbation move is used for diversification purposes.

Our VND algorithm uses three neighbourhood structures. The first one, called *Internal Swap*, attempts to replace a strategy \( j \) for a given edge \( i \) with another strategy \( j' \), of said edge, that is not contained in the tabu list and for which enough budget is available.

The second neighbourhood structure, called *External Swap*, attempts to replace one strategy \( j \) for edge \( i \), which will be assigned strategy \( j = 0 \), with another strategy \( j' \), which is not in the tabu list, associated to a different edge \( i' \). In practice, the security budget is transferred from edge \( i \) to edge \( i' \) with the cost of strategy \( j' \) lower than the remaining budget plus the cost of strategy \( j \).

The third neighbourhood structure, called *Double Swap*, is a variant of the first move. Two *Internal Swap* moves are executed at the same time. In practice, strategies \( j \) and \( j' \) associated to edges \( i \) and \( i' \) respectively
are simultaneously swapped with two other strategies $j$ and $j'$ available for edges $i$ and $i'$. The new strategies $j$ and $j'$ must not be present in the tabu list and the cost of the strategies $j$ and $j'$ must not exceed the remaining budget plus the cost of the removed strategies $j$ and $j'$.

To speed up the improvement stage, only a restricted number of edges, $\alpha$, are considered and only moves which have a positive contribution to the quality of the current solution are executed. In particular, the External Swap move attempts to replace a strategy $j$ associated to any edge $i$ already in the solution with one of the available strategies associated to the first $\alpha$ edges that do not have a security strategy in the solution. The candidate edges are ordered by decreasing value of criticality, as described before.

The size of the neighbourhood that Double Swap has to explore is equal to $\mathcal{E} \times (\mathcal{E} - 1)$ where $|\mathcal{E}|$ is the number of edges that have a security strategy in the current solution. For this reason, the number of evaluations that are performed by the Double Swap move is restricted to a maximum value based on a percentage of set $\mathcal{E}$, which is defined by the decision maker. The candidates for the Double Swap move are selected at random from all the edges that have a security strategy that is different from strategy 0.

Two different selection mechanisms can be used by the VND heuristic to improve the current solution. The first one is based on the first improvement mechanism, the second one is based on the best improvement mechanism. Both are explained in Section 2.4.1. The VND heuristic ends when no improvement of the current solution can be found.
2.4.3 Diversification phase

The diversification phase attempts to escape from the local optimum, reached in the improvement phase, by exploring different areas of the search space that are hopefully not yet explored. Two different diversification mechanisms are used in our variable neighbourhood tabu search heuristic: (1) A perturbation heuristic is used, as long as a maximum number of iterations without improvement of the current solution has not been reached; (2) A new initial solution is generated using the GRASP constructive heuristic, if the number of iterations without improvement becomes larger than a predefined maximum.

The perturbation heuristic partially destroys the current solution by setting the strategies that have been used for a number of edges to the zero, “do-nothing”, strategy. The strategies that have been removed are inserted in a tabu list in order to avoid their reuse in future solutions for a given number of iterations. The perturbation heuristic allows to free some budget resources, making room for new, unused strategies that are not in the tabu list. These new strategies are selected using either a best improvement or a first improvement selection mechanism (see Section 2.4.1) and added to the current solution as long as budget is not depleted and at least one strategy, with a cost less than the remaining budget and not contained in the tabu list, is available. The newly generated solution becomes the input of the VND heuristic for further improvement.

When the number of iterations without improvement becomes larger than a predefined maximum, a new solution is generated using the GRASP constructive heuristic.
2.5 Computational experiments

In this section, the different steps used to execute our computational experiments are described. First, a set of instances is generated. They are used in the second and third step for tuning and testing of the metaheuristic described before (Section 2.5.1).

In the second step, a statistical experiment is set up to determine the optimal parameter settings for the metaheuristic in order to achieve the best possible results (Section 2.5.2).

Finally, the metaheuristic with its best parameter settings is used to solve the test instances and computational results are shown in Section 2.5.3.

2.5.1 Instance generation

A library of instances has been created to test the algorithm developed in this chapter. All instances can be found on http://antor.uantwerpen.be/downloads/NS. The software that was developed to generate the instances, takes several parameters as input, necessary to specify the properties of the instances. These parameters are: (1) the maximum budget that can be spent on strategies; (2) the number of nodes in the network; (3) the maximum number of strategies for each edge; (4) the percentage of edges of the Delaunay triangulation (Delaunay, 1934) that should be added to the minimum spanning tree, to add some redundancy as explained below; (5) the number of instances to be generated for each set of parameters. As most of the information needed
for the creation of realistic instances is not available, some choices and assumptions have been made, to create what could be classified as realistic instances.

The number of scenarios contained in \( C \), the set of all critical scenarios, is dependent on the number of edges in and the topology of the network. In the worst case, the cardinality of this set is equal to \( 2^{|E|} \). This implies, e.g., that adding an edge to a network containing 20 edges, requires 1048576 \( (2^{21} - 2^{20}) \) more scenarios to be analysed and stored in memory. Due to the fact that the memory usage is doubled with every edge that is added to the instance, instances with 25 edges are the largest instances that could be solved by the desktop system used for the experiments. With the implementation of the algorithm used to conduct the experiments, instances with 25 edges consume over 8GB of memory. Instances with 26 edges consume more memory than is available in the machine that is used to run the tests on.

A first step in the instance generation process is the generation of nodes. Each node’s coordinates are generated by using a uniform distribution in the range from 0 to 100 for the x and y-coordinates. This range can be adjusted by the decision maker, but for the purpose of creating test instances are of no real importance, as the size or distance between nodes is never used. When the nodes have been generated, a Delaunay triangulation of this set is created. The Delaunay triangulation is used to identify the “neighbours” of each node. The connections between these neighbours will be used to create the final network. When the neighbours of all nodes are identified, Kruskal’s algorithm (Kruskal, 1956) is used to generate a minimum spanning tree. The motivation
for starting from a minimum spanning tree is that this is the most cost-effective way to connect all nodes in the network.

In real life cases, service providers often add redundancy to increase operational security. If one edge gets disabled, customers are serviced through the redundant edge. For this reason, the minimum spanning tree is complemented with edges to mimic the redundancy that is present in real life networks. This is done by selecting a number of edges from the Delaunay triangulation, which are not already used in the minimal spanning tree. The edge selection is based on their length, shortest first, and on a parameter that defines how many edges should be added, which is defined by the decision maker.

When the edges of the network have been generated, each edge is assigned a random probability of failure. These probabilities are different for each edge. This approach is adopted to make the instances more general. From a security related perspective, this can be justified by the difference in risks based on geographical location, length and criticality of that edge in the network. From the perspective of safety, it could be that certain sections in the network are added on a later point in time, which would yield a lower risk for failure due to the newer parts that were used, and the lower amount of normal wear and tear.

When each edge has been assigned a probability, the algorithm generates a random number of security strategies, with a minimum of one and a maximum number as defined in the parameters passed by the decision maker. Each strategy is assigned a random reduction in probability of failure between 1 and 20 percent, and a random cost based on this percentage reduction. To obtain this cost, and to have some
variation in the ratio between cost and the reduction, the percentage reduction is multiplied with a random number between 0.5 and 1.5 and the maximum budget. The maximum budget is used to have costs that are proportional to the available budget. The range of percentage reduction is selected on practical grounds. An improvement larger than 20 percent will, in practice, not be possible given the budget constraint.

The generated test instances’ names are encoded as follows:

NS-n8-c5-C3-a30-x0, where NS is referring to the network security problem, n represents the number of nodes in the instance, c gives the maximum number of security strategies generated for each edge, C is the maximum number of connections from and to a node (this is currently not used in the instance generator, but the option is foreseen in the encoding for future extensions), a represents the percentage of extra arcs from the Delaunay triangulation added to the minimal spanning tree, and finally, x represents the index number of the instance, which is particularly useful when multiple instances with the same settings are generated at once.

Twenty-six test instances were generated. The first set consists of 8 instances which have 8 nodes, 5 security strategies and have 30 percent of the edges of the Delaunay triangulation added to the minimal spanning tree. A second and third set, consisting of 8 and 10 instances respectively, have the same settings, but 9 and 10 nodes per instance. A final set is generated, which has 5 instances which consist out of 18 nodes with 10 security strategies per edge and also have 30 percent of the edges of the Delaunay triangulation added to the minimal spanning tree.
2.5.2 Parameter tuning

The metaheuristic, described in Section 2.4, uses some internal parameters that need to be tuned in order to have a good compromise between speed of the heuristic and quality of the solutions. The various internal parameters have been described in the aforementioned section and are summarized in Table 2.4 together with the values for which they are tested.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Values</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>max-iter</td>
<td>Number of restarts</td>
<td>50</td>
<td>1</td>
</tr>
<tr>
<td>perturb-percent</td>
<td>Percent of edges removed during the perturbation phase</td>
<td>10%, 30%, 50%, 70%</td>
<td>4</td>
</tr>
<tr>
<td>alpha</td>
<td>Size of the restricted candidate list</td>
<td>1, 2, 3, 4</td>
<td>4</td>
</tr>
<tr>
<td>tabu-tenure</td>
<td>Number of iterations that a strategy is kept in the tabu list</td>
<td>10, 30, 50</td>
<td>3</td>
</tr>
<tr>
<td>max-iter-no-improvement</td>
<td>Number of iterations without improvements</td>
<td>5, 10, 20</td>
<td>3</td>
</tr>
<tr>
<td>double-swap-percentage</td>
<td>Percentage value used to find the amount of evaluations to be performed by the Double swap move</td>
<td>20%, 50%, 80%</td>
<td>3</td>
</tr>
<tr>
<td>selection mechanism</td>
<td>Strategy to select edges</td>
<td>first, best</td>
<td>2</td>
</tr>
</tbody>
</table>

The parameter tuning experiment is conducted on a set of instances consisting of networks of 18 nodes and 20 edges.

A full factorial statistical experiment is carried out using the values re-
ported in Table 2.4. The objective values and times are averaged, and shown in Figure 2.4. The parameters with an asterisk had a significant effect on both the objective value and time, while parameter tabu-tenure was significant only in regard to the time. Parameter max-iter-no-improvement did not have a significant effect. A parameter setting equal to 10 was selected for this parameter. Even better solutions could be possibly found with parameter values outside of the range of tested values, especially as often the selected value for a parameter was at the end of a range of tested values. However, the solutions found by the algorithm with the selected parameter values were already of very good quality, and hence the choice was made to accept the end values of the proposed ranges as the selected values.

Figure 2.4: Plot of average objective values and times for the given parameter settings

Only taking into account the objective value for choosing the parameter settings, the selected settings for the metaheuristic parameters are shown in Table 2.5.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>max-iter</td>
<td>50</td>
</tr>
<tr>
<td>perturb-percent</td>
<td>10%</td>
</tr>
<tr>
<td>alpha</td>
<td>4</td>
</tr>
<tr>
<td>tabu-tenure</td>
<td>30</td>
</tr>
<tr>
<td>max-iter-no-improvement</td>
<td>10</td>
</tr>
<tr>
<td>double-swap-percentage</td>
<td>80%</td>
</tr>
<tr>
<td>selection mechanism</td>
<td>first</td>
</tr>
</tbody>
</table>

### 2.5.3 Computational results

After having selected the best parameter settings for the solution approach, the influence of each metaheuristic component on both the quality of the solutions and the running time is analysed. A set of instances ranging from 9 up to 20 edges and 5 up to 20 security strategies per edge is used for this purpose.

In order to analyse the quality of the solutions, the results found by the proposed metaheuristic are compared with the optimal solutions obtained by a naive exact approach. The exact approach is based on an exhaustive exploration of the search space, where the budget constraint is used as a cutting plane to speed up the running time. The term "naive" is referring to the fact that the implementation is done with a simple recursive code and no highly optimised solver like Gurobi or cplex are used. In Table 2.6 the results of and comparison between the metaheuristic and the exact approach are reported for instances with 8 nodes and 9 edges (instance names with the code n8) and with 9 nodes and 11 edges (instance names with the code n9) and 5 security strategies per edge. The metaheuristic was run 25 times on each instance to com-
pute the average running time and objective value that are reported in Table 2.6 together with the best solution found over these runs. The metaheuristic obtained 9 optimal solutions (highlighted in bold in column Best Gap) out of 16 test instances. The percentage Best gap, used as an indicator of the quality of the obtained solutions, is on average around 0.53%. Considering all the solutions averaged over 25 runs the average gap from the optimal solutions is only 1.59%.

The CPU times needed by the exact approach, increase exponentially with the number of edges in the network, while the CPU time required by the metaheuristic remains stable. Analysing both the exact approach and the metaheuristic, it becomes clear that the proposed metaheuristic achieves good results in a much shorter amount of time.

For larger instances, CPU times of the exact approach quickly become intractable. For an instance with 13 edges, the exact approach took around 3 days to find the optimal solution. The heuristic solution was only 1.65% from this optimal value, and was found after 0.475 seconds. A time limit of 5 hours was imposed on the exact solution method to find a good upper-bound. This value is used in Table 2.7 to evaluate the quality of the results provided by the metaheuristic.

It is worth noticing that the solution obtained by using a time limit for the instance NS-n10-c5-C3-a30-x2 (shown in Table 2.7) resulted in a good upper bound, presenting only a difference of 0.11% from the optimal known solution. This means that the selected time limit is adequate, for that instance, to achieve a good upper-bound to be used to analyse the quality of the solutions obtained by the metaheuristic.
Table 2.6: Results of the exact approach in comparison with the metaheuristic for different instances

<table>
<thead>
<tr>
<th>Instance</th>
<th>Exact Approach</th>
<th>Metaheuristic</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Optimal Solution</td>
<td>Time (s)</td>
<td>Best Solution</td>
<td>Average Solution</td>
<td>Average Time (s)</td>
<td>Best Gap (%)</td>
<td>Average Gap (%)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NS-n8-c5-C3-a30-x0</td>
<td>0.648</td>
<td>637</td>
<td>0.652</td>
<td>0.655</td>
<td>0.147</td>
<td>0.57</td>
<td>1.01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NS-n8-c5-C3-a30-x1</td>
<td>0.814</td>
<td>37</td>
<td>0.814</td>
<td>0.844</td>
<td>0.614</td>
<td>0.00</td>
<td>3.66</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NS-n8-c5-C3-a30-x2</td>
<td>0.182</td>
<td>104</td>
<td>0.185</td>
<td>0.185</td>
<td>0.642</td>
<td>1.71</td>
<td>2.08</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NS-n8-c5-C3-a30-x3</td>
<td>0.134</td>
<td>58</td>
<td>0.137</td>
<td>0.143</td>
<td>0.573</td>
<td>2.24</td>
<td>6.71</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NS-n8-c5-C3-a30-x4</td>
<td>0.648</td>
<td>55</td>
<td>0.654</td>
<td>0.664</td>
<td>0.439</td>
<td>0.87</td>
<td>2.45</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NS-n8-c5-C3-a30-x5</td>
<td>0.341</td>
<td>14</td>
<td>0.341</td>
<td>0.343</td>
<td>0.504</td>
<td>0.00</td>
<td>0.80</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NS-n8-c5-C3-a30-x6</td>
<td>0.252</td>
<td>3</td>
<td>0.258</td>
<td>0.262</td>
<td>0.697</td>
<td>2.54</td>
<td>3.91</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NS-n8-c5-C3-a30-x7</td>
<td>0.079</td>
<td>66</td>
<td>0.079</td>
<td>0.079</td>
<td>0.549</td>
<td>0.00</td>
<td>0.54</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NS-n9-c5-C3-a30-x0</td>
<td>0.893</td>
<td>1357</td>
<td>0.894</td>
<td>0.898</td>
<td>0.527</td>
<td>0.57</td>
<td>0.12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NS-n9-c5-C3-a30-x1</td>
<td>0.821</td>
<td>5744</td>
<td>0.821</td>
<td>0.821</td>
<td>0.702</td>
<td>0.00</td>
<td>0.12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NS-n9-c5-C3-a30-x10</td>
<td>0.412</td>
<td>223</td>
<td>0.412</td>
<td>0.423</td>
<td>0.218</td>
<td>0.00</td>
<td>2.78</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NS-n9-c5-C3-a30-x11</td>
<td>0.073</td>
<td>589</td>
<td>0.073</td>
<td>0.073</td>
<td>0.254</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NS-n9-c5-C3-a30-x12</td>
<td>0.797</td>
<td>1160</td>
<td>0.797</td>
<td>0.798</td>
<td>0.414</td>
<td>0.00</td>
<td>0.07</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NS-n9-c5-C3-a30-x14</td>
<td>0.111</td>
<td>7826</td>
<td>0.111</td>
<td>0.111</td>
<td>0.503</td>
<td>0.00</td>
<td>0.16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NS-n9-c5-C3-a30-x4</td>
<td>0.408</td>
<td>3178</td>
<td>0.408</td>
<td>0.410</td>
<td>0.304</td>
<td>0.00</td>
<td>0.34</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NS-n9-c5-C3-a30-x8</td>
<td>0.431</td>
<td>8752</td>
<td>0.431</td>
<td>0.433</td>
<td>0.594</td>
<td>0.04</td>
<td>0.67</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>0.440</strong></td>
<td><strong>1862.688</strong></td>
<td><strong>0.442</strong></td>
<td><strong>0.446</strong></td>
<td><strong>0.480</strong></td>
<td><strong>0.534</strong></td>
<td><strong>1.589</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The results are shown in Table 2.7, where on average the metaheuristic improved the level of security of the initial network by 32% and the Upper-bound for the exact approach by 4.4%. In eight cases, the metaheuristic outperformed the exact approach with the time limit, finding better results in a shorter computational time. An increase in CPU time from less than one second to up to 438 seconds is observed with the increase of nodes and edges in the instances. This is to be expected, as the number of critical scenario’s to evaluate in the worst case is $2^{10}$ times higher, and each edge has twice the amount of available security strategies.
Table 2.7: Results for instances in the range 10-20 edges and 5-20 security strategies per edge

<table>
<thead>
<tr>
<th>Instance</th>
<th>Exact Approach</th>
<th>Metaheuristic</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Original risk</td>
<td>Upper-bound</td>
<td>Time (s)</td>
<td>Best Solution</td>
<td>Average Solution</td>
<td>Time (s)</td>
<td>Best Gap (%)</td>
<td>Average Gap (%)</td>
</tr>
<tr>
<td>NS-n10-c5-C3-a30-x0</td>
<td>≈1</td>
<td>0.918</td>
<td>18000</td>
<td>0.913</td>
<td>0.913</td>
<td>0.614</td>
<td>-0.61</td>
<td>-0.56</td>
</tr>
<tr>
<td>NS-n10-c5-C3-a30-x1</td>
<td>≈1</td>
<td>0.201</td>
<td>18000</td>
<td>0.167</td>
<td>0.167</td>
<td>0.374</td>
<td>-17.21</td>
<td>-16.76</td>
</tr>
<tr>
<td>NS-n10-c5-C3-a30-x2*</td>
<td>≈1</td>
<td>0.861</td>
<td>18000</td>
<td>0.876</td>
<td>0.887</td>
<td>0.475</td>
<td>1.65</td>
<td>3.04</td>
</tr>
<tr>
<td>NS-n10-c5-C3-a30-x3</td>
<td>≈1</td>
<td>0.581</td>
<td>18000</td>
<td>0.552</td>
<td>0.557</td>
<td>0.453</td>
<td>-5.03</td>
<td>-4.24</td>
</tr>
<tr>
<td>NS-n10-c5-C3-a30-x4</td>
<td>≈1</td>
<td>0.645</td>
<td>18000</td>
<td>0.647</td>
<td>0.671</td>
<td>0.353</td>
<td>0.29</td>
<td>4.03</td>
</tr>
<tr>
<td>NS-n18-c10-C3-a30-x0</td>
<td>≈1</td>
<td>0.997</td>
<td>18000</td>
<td>0.985</td>
<td>0.985</td>
<td>438</td>
<td>-1.22</td>
<td>-1.22</td>
</tr>
<tr>
<td>NS-n18-c10-C3-a30-x1</td>
<td>≈1</td>
<td>0.616</td>
<td>18000</td>
<td>0.558</td>
<td>0.558</td>
<td>208</td>
<td>-9.44</td>
<td>-9.43</td>
</tr>
<tr>
<td>NS-n18-c10-C3-a30-x2</td>
<td>≈1</td>
<td>0.946</td>
<td>18000</td>
<td>0.901</td>
<td>0.901</td>
<td>364</td>
<td>-4.79</td>
<td>-4.78</td>
</tr>
<tr>
<td>NS-n18-c10-C3-a30-x3</td>
<td>≈1</td>
<td>0.407</td>
<td>18000</td>
<td>0.312</td>
<td>0.312</td>
<td>344</td>
<td>-23.43</td>
<td>-23.41</td>
</tr>
<tr>
<td>NS-n18-c10-C3-a30-x4</td>
<td>≈1</td>
<td>0.935</td>
<td>18000</td>
<td>0.844</td>
<td>0.845</td>
<td>255</td>
<td>-9.65</td>
<td>-9.63</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>≈1</td>
<td>0.711</td>
<td>18000</td>
<td>0.676</td>
<td>0.680</td>
<td>161.127</td>
<td>-6.944</td>
<td>-6.296</td>
</tr>
</tbody>
</table>

Figure 2.5 reports the evolution of the objective value associated with both the best and the current solutions over time. It can be noticed that the proposed solution approach is able to converge towards good results in a short CPU time also in case of large instances. The marginal improvement of the best solution found so far significantly decreases when the running time is increased.

Analysing the plot associated with the current solution in Figure 2.5 clarifies the behaviour of the solution approach. After the perturbation, that destroys the quality of the solution, small improvements obtained during the VND heuristic can lead to better solutions. One can clearly distinguish the perturbation strategy that allows the algorithm to efficiently escape from local optima. Starting from the perturbed solution, denoted with a peak in the graph in Figure 2.5, the VND heuristic guides the current solution through small improvements towards a new local optimum and hopefully a new and better solution. The fact that the
VND heuristic is able to decrease the value of the perturbed solution and detect new local optima proves the efficiency of the VND.

The effect of each metaheuristic component on the objective value is also analysed. The VND heuristic on average can improve the initial solutions generated by the GRASP constructive heuristic by 1%. This value is low because the GRASP heuristic can already find very good results.

### 2.6 Alternative model

The model and associated solution approach, proposed in this chapter, could be substituted with a different one, that achieves the same end result. The main difference, between the previous solution approach and the one introduced here, is the manner in which the probability that no path exists between node $o$ and node $d$ is calculated.
In Equation (2.4) the mathematical formulation of the probability of a certain scenario $l$ is defined as the product of the probabilities of failure for all edges that fail in said scenario, multiplied with the product of the probabilities of not failing for all the edges that do not fail in scenario $l$. The events of failure are independent of each other. The result of each scenario is stored in a list. This list can grow very big because of the number of scenarios, which poses a problem to computers with limited memory capacity. An alternative would be to use minimal cut sets to achieve the probability of disconnection between node $o$ and node $d$.

A minimal cut set is defined as a set of failing edges that disconnect every path between node $o$ and node $d$, for which if any of the edges is removed from this set (i.e., is not failing), the condition that node $o$ is disconnected from node $d$ no longer holds true. Each scenario is represented by a cut set, while a cut set can represent multiple scenarios.

As for the cut sets only the probability of the cut happening is of importance, and the information that an event takes place in any of the edges that are not part of the cut set is of no importance, the formula to calculate the probability for that cut can be adapted from Equation (2.4). The second part of the formula, where the product of the probabilities of not failing for all the edges that do not fail is calculated, can be removed. We will only take the product of the probability for failure of the edges that are part of the cut set.

To find the total probability of disconnection in the whole graph, each combination of cuts should be calculated. However, when calculating the probability of the combinations of cuts, special attention should be
given to the fact that the same edge can be part of multiple cuts, and should be only used once in the end probability.

When the probabilities of all the combinations of cuts are known, the probabilities of combinations of cuts with an odd number of cuts should be summed, while probabilities of the combinations with an even number of cuts should get subtracted.

Cuts can have a huge positive or negative impact on the computational time and memory requirements in comparison to the previous approach. It is highly dependent on the topology of the graph. Some examples and corresponding number of combinations and cut sets are shown in Figure 2.6 and Table 2.8

The main advantage of working with this approach over the approach where critical scenarios are used, is that for the approach with the minimal cut sets, we can postpone the calculation of the combinations of minimal cut sets and of the total probability of disconnection to the
2.6 Alternative model

Table 2.8: Properties of the graphs shown in Figure 2.6

<table>
<thead>
<tr>
<th>Subfigure</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>edges</td>
<td>5</td>
<td>11</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>cuts</td>
<td>5</td>
<td>1</td>
<td>25</td>
<td>16</td>
</tr>
<tr>
<td>scenarios</td>
<td>31</td>
<td>2047</td>
<td>1023</td>
<td>$2^{15} - 1$</td>
</tr>
<tr>
<td>critical scenarios</td>
<td>31</td>
<td>1024</td>
<td>961</td>
<td>2504</td>
</tr>
<tr>
<td>cut combinations</td>
<td>31</td>
<td>1</td>
<td>$2^{25} - 1$</td>
<td>$2^{16} - 1$</td>
</tr>
</tbody>
</table>

end of the algorithm. The sum of the probabilities of the minimal cut sets can be used as an estimate of the total probability throughout the algorithm. This means that the number of updates for the probabilities when a security strategy is applied, can be reduced to the number of minimal cut sets, instead of the number of critical combinations. From Table 2.8, we can see that for the example graphs in Figure 2.6 in all cases there were less minimal cut sets than there were critical scenarios. For the approach with the minimal cut sets, the final probability calculation takes a bit longer, but this loss in time does not outweigh the savings in time generated during the heuristic algorithm.

Table 2.9 gives a comparison of the two approaches discussed in this chapter. The approach using the minimal cut sets always finds the same or an extremely similar solution in a much lower computational time. All the instances were ran 5 times on the same machine for each algorithm, and the best and average values are shown in the table, as well as the gap between the best found and average found objective values.
Table 2.9: Results for the metaheuristic using critical scenarios and minimal cut sets to calculate the objective value

<table>
<thead>
<tr>
<th>Instance</th>
<th>Scenarios</th>
<th>Cut sets</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Best</td>
<td>Average</td>
<td>Time</td>
<td>Best</td>
<td>Average</td>
</tr>
<tr>
<td></td>
<td>Solution</td>
<td>Solution</td>
<td>(s)</td>
<td>Solution</td>
<td>Solution</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NS-n18-c10-C3-a30-x0</td>
<td>0.985</td>
<td>0.985</td>
<td>516</td>
<td>0.985</td>
<td>0.985</td>
</tr>
<tr>
<td>NS-n18-c10-C3-a30-x1</td>
<td>0.558</td>
<td>0.558</td>
<td>253</td>
<td>0.558</td>
<td>0.558</td>
</tr>
<tr>
<td>NS-n18-c10-C3-a30-x2</td>
<td>0.901</td>
<td>0.901</td>
<td>432</td>
<td>0.901</td>
<td>0.901</td>
</tr>
<tr>
<td>NS-n18-c10-C3-a30-x3</td>
<td>0.312</td>
<td>0.312</td>
<td>388</td>
<td>0.312</td>
<td>0.312</td>
</tr>
<tr>
<td>NS-n18-c10-C3-a30-x4</td>
<td>0.845</td>
<td>0.845</td>
<td>356</td>
<td>0.845</td>
<td>0.845</td>
</tr>
</tbody>
</table>

The difficulty of the approach discussed here, is that finding the minimal cut sets in itself is NP-hard (Benaddy and Wakrim, 2012). The generation of these minimal cut sets falls out of the scope of this work, and the reader is referred to Tsukiyama et al. (1980) and Benaddy and Wakrim (2012) for possible approaches to enumerate them efficiently. The algorithm developed in this chapter, uses a naive enumeration approach to generate all minimal cut sets. The changes to the rest of the heuristic as described in previous sections are limited to the calculation of the objective value only.

2.7 Conclusion

In this chapter, a model for the selection of the appropriate security strategies, given a limited budget, is proposed to increase the security of a network infrastructure such as pipeline transportation systems, telecommunication networks and smart grids.
By selecting one origin node and one destination node, it is possible to define the risk of an interruption of service (or material flow) due to component failure by external malicious attacks or safety related reasons.

An exact evaluation of the risk of the whole network being unavailable might be a difficult task, especially when several loops are present inside the network. In order to reduce the complexity of computations, a heuristic approach to have an accurate estimate of the risk of the network being down is defined.

The variable neighbourhood tabu search heuristic has been tuned in order to find its best configuration when minimising the objective value. In a second stage of the computational experiments, the tuned meta-heuristic has been employed to solve a set of test instances that mimic possible realistic scenarios and the performance of the algorithm is evaluated.

The variable neighbourhood tabu search heuristic can find very good results in a really short amount of time. It is performing extremely well in comparison to the naive exact approach. However, due to the fact that we need to generate and evaluate all scenarios to find the critical scenarios and the sheer number of critical scenarios, to tackle larger instances, a different approach is presented (see Section 2.6).

The next chapter will be aimed at extending the model to support more realistic scenarios where e.g., the nodes are vulnerable and the connection of all nodes in the network should be ensured.
Models and algorithms for the optimisation of security strategies in networks with multiple source and destination nodes

This chapter is based on the manuscript “Janssens, J., Talarico, L., and Sörensen, K. (2016d). A metaheuristic for security budget allocation in utility networks. International Transactions in Operational Research. Accepted for publication”
3.1 Introduction

As described in Chapter 2, utility networks are subject to failure due to different types of causes. The main goal of that chapter was to develop a solution method that would guide decision makers in making informed decisions about how to allocate a security budget into different security strategies, to reduce the risk of a network interruption. To do so, a few assumptions were made, to simplify the problem considerably. The downside of these assumptions, is that an abstraction of reality was made. To ameliorate that simplification, in order to get to a more usable and more generally applicable approach, a more elaborate model should be created, and the algorithm should be adapted accordingly.

The first assumption, made in Chapter 2, is that only edges are subject to failure. In a network, due to the abundant length of pipes or cables and the distances over which they are spread out, they are a viable and easy target of possible attacks. However, the impact on the network is rather small, and repairs are fairly easy. If an attacker would like to make a greater impact on a network, connection points, like substations and switching boards, present a more rewarding target, as they tend to disrupt the service in a larger part of the network. In terms of an electricity network, power generating stations are the most difficult targets for an attack, since they are usually guarded and protected more effectively. The impact of such type of attacks is even bigger than the ones discussed before. In 2014, a sabotage action in a Belgian nuclear power plant, causing it to shut down for several months, proves that
even well guarded places with an ample amount of security and safety regulations can be a viable target.

The second assumption made in Chapter 2, is the consideration of only one source node and one destination node. In this chapter, this is extended in such a way that a service interruption is defined as a disconnection between any two nodes in the network.

Hence, the goal of this chapter, similar to the previous one, is to develop a decision model to reduce the vulnerability of a utility network by allocating a security budget amongst different security strategies. In the model, designed in this chapter, each segment (edge or node) of the network has a certain probability to be targeted by intentional attackers and failing as a result of that attack. It is assumed that these probabilities are known or can be estimated. References to research that discuss risk assessment techniques are listed in the previous chapter. The manager of the utility network can implement security strategies on each segment, which reduce the likelihood of that segment to fail. Each security strategy has a certain cost and impact, and the manager has a budget that cannot be exceeded.

For a literature overview, the reader is referred to Section 2.2. The remainder of this chapter is organized as follows. In Section 3.2, the problem of selecting the best strategies to increase the security of the whole network is described and modelled as an optimization problem. Section 3.3, proposes a metaheuristic to solve the network security problem. Section 3.4 presents the results of the heuristic on realistic instances. Section 3.5 discusses the main research findings and concludes the chapter.
3.2 Problem description

3.2.1 Edge and node failures

A utility network can be represented by a graph $G = \{N, E\}$, where $N$ represents the set of nodes and $E$ the set of edges. All edges $i \in E$ and nodes $k \in N$ have a probability of failure, denoted as $p_e^i$ and $p_n^k$, where the index $e$ refers to edges and index $n$ is used for the nodes. A node or edge failure always completely disables the respective node or edge.

A set of security strategies $S_e^i$ and $S_n^k$, is defined for each edge $i \in E$ and each node $k \in N$. For each security strategy $j \in S_e^i$ (or $S_n^k$) of edge $i$ (node $k$) there is a cost $c_e^{ij}$ ($c_n^{kj}$), and a value $p_e^{ij}$ ($p_n^{kj}$), which is the probability of a failure of edge $i$ (node $k$) when this security strategy is applied. A single security strategy needs to be selected for each node and each edge in the graph. The probability $p_e^i$ of edge $i$ failing and $p_n^k$ of node $k$ failing after applying a security strategy $j$ will be equal to the probability $p_e^{ij}$ or $p_n^{kj}$ that is associated to that security strategy.

In this chapter, the assumption is made that a preliminary risk assessment phase has been conducted by experts, in order to determine the probability of failure associated with each edge or node, together with the costs and benefits of each available security measure.

When a node fails, this is equivalent to the failure of all the incident edges. In other words, if a node is not available due to a failure, the edges entering in and leaving from that node need to be considered out of service. In the calculations of the index of connectivity a failure
Chapter 3 Multiple source and destination security strategy optimisation

(a) Attack on a node  
(b) Equivalent failure of edges

Figure 3.1: Substitution of a failure of a node (a) with the failure of edges (b) for the computation of the connectivity index of a node is treated as the failure of the edges that enter or leave from the node. An example is shown in Figure 3.1

In Chapter 2, a security strategy can be a combination of several individual security measures. (see e.g., Table 2.1). A combination of security measures can have a different effectiveness than the sum of the impact of the individual security measures due to some interaction effects. In some cases, combinations of single security measures might not be possible due to their incompatibility.

Also in this chapter, the default security strategy (labelled 0) for edge $i$, that has a cost $c^{e}_{i0} = 0$, is a base case that indicates that no security measure is applied. Its related probability $p^{e}_{i0}$ represents the probability of a failure of edge $i$ in case no security strategy is selected. This also applies to the security strategies and probabilities for the nodes.
3.2 Problem description

3.2.2 Mathematical formulation

The model defined in this chapter selects a security strategy for each edge and each node such that the total risk of network failure is minimized, and a budget constraint is satisfied. In the previous chapter, we have developed a simplified model with a given origin node $o$ and a destination node $d$. In that model, a network failure occurs if these two nodes are disconnected, i.e., if no functioning path between $o$ and $d$ exists, where a functioning path is defined as a path that does not contain any failed nodes or edges. In case of a network failure, it is impossible for a service or good from node $o$ to reach node $d$ (e.g., it would be impossible to make a phone call from node $o$ to node $d$).

Here, we develop an extension of this basic model. The changes are twofold. Firstly, failures in nodes are also considered, as opposed to the simplified model, where only edges could fail. Secondly, a network failure is said to occur if any pair of nodes is disconnected, i.e., if two nodes exist in the graph such that no functioning path has these nodes as endpoints, while in the simple model only one origin and one destination node were considered.

The model developed here minimizes the probability that a network failure will occur. To calculate the probability of network failure, a list of all critical scenarios is created. A scenario contains the state (failed or functioning) for each edge and each node. A critical scenario is a scenario that causes a network failure. We say that each element $l$ of set $C$, the set of critical scenarios, is composed of edges and nodes that fail (set $E^F_l$ and $N^F_l$ respectively), and edges and nodes that do not fail
(sets $\mathcal{E}_l^N$ and $\mathcal{N}_l^N$ respectively). It should be noted that $\mathcal{E}_l^F \cup \mathcal{E}_l^N \cup \mathcal{N}_l^F \cup \mathcal{N}_l^N = \mathcal{E} \cup \mathcal{N}$, $\forall l \in C$. Given the failure probabilities for each edge and each node (which depend on the security strategies selected for these edges and nodes), and given the state of each edge and node in scenario $l$, the probability of occurrence of this scenario $l$, which we denote $R_l$, can be calculated as the product of the probabilities for each edge and node to be in their given state.

\[
R_l = \prod_{i \in \mathcal{E}_l^F} p_i^e \cdot \prod_{i \in \mathcal{E}_l^N} (1 - p_i^e) \cdot \prod_{k \in \mathcal{N}_l^F} p_k^n \cdot \prod_{k \in \mathcal{N}_l^N} (1 - p_k^n) \quad (3.1)
\]

Because scenarios are mutually exclusive events, the total probability of network failure (i.e., the objective function of the model developed in this chapter), can be calculated by summing the probabilities of all critical events, i.e., $\sum_{l \in C} R_l$.

Obviously, the cardinality of $C$ depends on the topology of $G$, but will be very large in many situations. In this chapter, we calculate the set $C$ by checking all possible scenarios. Because each edge and each node can be in exactly two states (functioning or failed), the number of possible scenarios, each of which has to be checked for criticality, is $2^{|\mathcal{E}| + |\mathcal{N}|}$. Checking a scenario for criticality is equivalent to checking whether the graph in which the edges and nodes have been removed that fail in the scenario, is connected. To verify this, a simple breadth-first search algorithm is used.

Similar to the previous chapter, in the model defined here, a budget constraint limits the security strategies that can be selected. Let $B$ represent the available security budget and let $x_{ij}^e$ be a binary variable that
takes value 1 when security strategy \( j \) on edge \( i \) is applied, and 0 otherwise. Let \( x^n_{kj} \) be a binary variable that takes value 1 when security strategy \( j \) on node \( k \) is applied, and 0 otherwise. As mentioned, set \( S^e_i \) includes all the security strategies \( j \) for edge \( i \) with \( j = 0 \) being the situation in which no security measures for edge \( i \) are applied. The same applies for set \( S^n_k \) for all nodes \( k \in \mathcal{N} \). A mathematical model to select the optimal security strategy for each edge and each node is the following.

\[
\min \sum_{l \in \mathcal{C}} R_l \tag{3.2}
\]

s.t.

\[
\sum_{i \in \mathcal{E}} \sum_{j \in S^e_i} c^e_{ij} \cdot x^e_{ij} + \sum_{k \in \mathcal{N}} \sum_{j \in S^n_k} c^n_{kj} \cdot x^n_{kj} \leq B \tag{3.3}
\]

\[
p^e_i = \sum_{j \in S^e_i} p^e_{ij} \cdot x^e_{ij} \quad \forall i \in \mathcal{E} \tag{3.4}
\]

\[
p^n_k = \sum_{j \in S^n_k} p^n_{kj} \cdot x^n_{kj} \quad \forall k \in \mathcal{N} \tag{3.5}
\]

\[
R_l = \prod_{i \in \mathcal{E}^l_i} p^e_i \cdot \prod_{i \in \mathcal{N}^l_i} (1 - p^e_i) \cdot \prod_{k \in \mathcal{N}^l_i} p^n_k \cdot \prod_{k \in \mathcal{N}^l_i} (1 - p^n_k) \quad \forall l \in \mathcal{C} \tag{3.6}
\]

\[
\sum_{j \in S^e_i} x^e_{ij} = 1 \quad \forall i \in \mathcal{E} \tag{3.7}
\]

\[
\sum_{j \in S^n_k} x^n_{kj} = 1 \quad \forall k \in \mathcal{N} \tag{3.8}
\]

\[
x^e_{ij} \in \{0, 1\} \quad \forall i \in \mathcal{E}, \forall j \in S^e_i \tag{3.9}
\]

\[
x^n_{kj} \in \{0, 1\} \quad \forall k \in \mathcal{N}, \forall j \in S^n_k \tag{3.10}
\]
The objective function in Equation (3.2) minimizes the total probability of network failure, and is calculated as the sum of probabilities of all critical scenarios. Constraint (3.3) ensures that the total cost associated to the selected security strategies does not exceed the predefined security budget $B$. Equations (3.4) and (3.5) are used to determine the probability $p_i^e$ of failure of edge $i$ and the probability $p_k^n$ of a failure of node $k$. Equation (3.6) calculates the probabilities of occurrence for all critical scenarios. Equations (3.7) and (3.8) force the decision process to the selection of exactly one security strategy for each edge or node, where $x_{i0}^e = 1$ or $x_{k0}^n = 1$ indicates that no security strategy (or security strategy 0) has been selected for edge $i$ and node $k$ respectively. Finally, Equations (3.9) and (3.10) enforce the domain of the decision variables, and ensure that no partial security strategies are allowed.

### 3.3 Solution approach

#### 3.3.1 Auxiliary calculations

The model presented in the previous section is computationally expensive because of the number of scenarios, and the resulting risk calculations. If the security strategy of a single node or edge changes, its failure probability changes and the probability of occurrence of every critical scenario is affected. This can be seen in Equation (3.6), in which the failure probabilities of each edge and node appear. Fortunately, it is not necessary to recalculate Equation (3.6) from scratch every time the security strategy of an edge or node changes.
The implementation of the proposed algorithm uses an update formula to recalculate the risk for each scenario when a security strategy is changed.

\[
\bar{R}_l = \frac{R_l \cdot \bar{P}}{P}
\]  

(3.11)

The formula divides the previous probability for the scenario, \( R_l \), with the previous failure probability of the edge or node \( P \), and multiplies it with the new probability, \( \bar{P} \), to obtain the new probability of the scenario, \( \bar{R}_l \). If the edge or node is failing in that scenario, \( P \) is substituted with \( p_x \), and \((1 - p_x)\) otherwise. This only requires two operations per scenario (one multiplication, one division), whereas recalculating Equation (3.6) from scratch results in \(|\mathcal{A}| + |\mathcal{N}|\) calculations per scenario.

To deduce this update formula, we start from Equation (3.6), which states the formula to calculate \( R_l \) as a product of the probabilities of all failing edges and nodes, multiplied with the product of 1 minus the probability of the non-failing edges and nodes. For a given scenario \( l \), if all edges and nodes except for one edge (or node) \( i \) stay the same, we could reformulate Equation (3.6) as follows:

\[
R_l = P_{\text{fix}} \cdot P
\]  

(3.12)

which could be rewritten as:

\[
P_{\text{fix}} = \frac{R_l}{P}
\]  

(3.13)
Where $P_{\text{fix}}$ is the product of all the unchanged probabilities and respective 1 minus probabilities, and $P$ is substituted with $p_i$ or $(1 - p_i)$, depending on the fact if edge (or node) $i$ is failing or not, in scenario $l$. Given a change in the probability of edge (node) $i$ which results in $\bar{P}$, we could write the following:

$$\frac{R_l}{P} = P_{\text{fix}} = \frac{\bar{R}_l}{\bar{P}}$$

(3.14)

3.3.2 Exact approach

To establish a benchmark for the metaheuristic developed in Section 3.3.3, we have implemented a naive exact approach that loops over each combination of security strategies and validates them for the budget constraint. If the combination of security strategies does not infringe the budget constraint, the exact approach verifies if it has a better result than the previous best found solution, and stores the new solution if it has.

Exact approaches to solve the problem, described in Equations (3.2) to (3.10), are viable methods for small instances only, due to the exponential nature of the number of critical combinations to be considered. In the worst case, the number of critical combinations that have to be analysed and updated through the solution process is equal to $2^{|\mathcal{A}|+|\mathcal{N}|}$.

The maximum number of possible ways to combine security strategies is equal to $\prod_{i \in \mathcal{A}} |S_i^a| \cdot \prod_{k \in \mathcal{N}} |S_k^n|$. In case the number of security strategies is equal for all nodes and edges and simplified as $|\mathcal{S}|$ this value can be rewritten as $|\mathcal{S}|^{|\mathcal{A}|+|\mathcal{N}|}$. As an illustration of the vast
3.3 Solution approach

![Graph showing relationship between number of edges and CPU time](image)

Figure 3.2: Relationship between the number of edges and the CPU time needed for the exact algorithm

Given the computational complexity of the problem, it is unlikely that an efficient exact approach can be developed. Therefore, we develop an efficient metaheuristic approach, sacrificing a guarantee of optimality in favour of finding near-optimal solutions in short running times.

Our algorithm is an iterated local search (ILS) (Lourenço et al., 2010) which is hybridised with a greedy randomized adaptive search procedure (GRASP) and a variable neighbourhood descent (VND) improvement heuristic. The algorithm always continues with the current solution. Two perturbation heuristics are used to escape from local optima. In addition, a tabu list is used during the whole execution of the heuristic to avoid an exploration of solutions that have been analysed in
Algorithm 3.1 Metaheuristic structure

1: Initialise both Problem and Heuristic parameters
2: $x \leftarrow \text{GRASP}()$
3: $x^* \leftarrow x, f(x^*) \leftarrow f(x)$
4: $\text{max-iter-no-improvement} \leftarrow 0$
5: $\text{iterations} \leftarrow 0$
6: while ($\text{iterations} < \text{max-iter}$) do
7: \hspace{1em} $x \leftarrow \text{VND}(x)$
8: \hspace{1em} if ($f(x) < f(x^*)$) then
9: \hspace{2em} $x^* \leftarrow x, f(x^*) \leftarrow f(x)$
10: \hspace{2em} $\text{max-iter-no-improvement} \leftarrow 0$
11: \hspace{1em} else
12: \hspace{2em} $\text{max-iter-no-improvement} \leftarrow \text{max-iter-no-improvement} + 1$
13: \hspace{1em} end if
14: \hspace{1em} if ($\text{max-iter-no-improvement} < \text{MaxPerturbations}$) then
15: \hspace{2em} $x \leftarrow \text{Perturbation}(x)$
16: \hspace{1em} else
17: \hspace{2em} $x \leftarrow \text{GRASP}()$
18: \hspace{1em} end if
19: \hspace{1em} $\text{iterations} \leftarrow \text{iterations} + 1$
20: end while
21: return $x^*$

previous iterations. Pseudo-code of the algorithm can be found in Algorithm 3.1, which is the same as Algorithm 2.1, with small changes to the calculation of the objective value and the data structures to hold the instance and solution.

An initialisation step is performed, followed by a construction phase, in which a GRASP heuristic is used to construct an initial solution. The solution found is then passed to a variable neighbourhood descent
(VND), which performs different moves on the solution, until no more improvement can be found, after which a differentiation phase is executed.

After the initialization step, an initial solution is generated by a GRASP constructive heuristic. The GRASP heuristic starts from a solution in which all edges and nodes have the default security strategy (strategy 0). The total cost of such a solution is zero. One step at a time, the GRASP heuristics upgrades security strategies for edges and nodes. The GRASP heuristic repeats the selection of an edge or node and an upgraded security strategy for that edge or node until the security budget does not allow any further security strategy upgrades. The selection of the edge or node to upgrade the security strategy for, happens by selecting a random edge or node from a restricted candidate list (RCL).

The RCL is created by sorting all nodes and edges by their “potential” to reduce the probability of network failure, and selecting the top alpha edges/nodes from that list. This potential is based on the frequency of occurrence of edges and nodes in critical scenarios, the probability of occurrence of that scenario, and the number of disconnections caused in the network. If an edge or node is part of many critical scenarios (in a failed state), it is likely to have a higher potential to improve the network failure probability when their own failure probability is decreased. Edges or nodes that (in a failed state) do not appear in many critical scenarios, are similarly unlikely to improve the network failure probability.

A first and best improvement strategy are implemented in the GRASP heuristic, and based on a parameter, one of them is used. For the best
improvement strategy, a restricted candidate list of edges and nodes is created and all possible security upgrades for the selected edge or node are evaluated. The security strategy that yields the highest improvement of the objective value is selected and applied to the designated edge or node. For the first improving strategy, a security strategy is selected at random from the whole list of security strategies for a random edge or node. If, at the end of the algorithm, for the first improvement strategy the randomly selected security strategy is out of budget, another security strategy is selected at random for that edge or node, and evaluated for the budget, until a solution strategy can be used as an upgrade, or all security strategies for that edge or node are evaluated. If no valid security strategy can be found for the selected edge or node, a new random edge or node is selected from the RCL.

The calculation of the potential for each edge and node is done by a method that ranks the set of critical combinations, $C$, based on an index of connectivity for each scenario $\text{Con}_l$. Given a scenario $l$, this index of connectivity can be calculated as the fraction of node pairs between which a functional path in the graph exists. Given a graph with $n$ nodes, where $n = |\mathcal{N}|$ the index of connectivity $\text{Con}_l$, is calculated as follows:

$$\text{Con}_l = \frac{\sum_{o=1}^{n} \sum_{d=1}^{n} r^l_{od}}{n \cdot (n - 1)},$$

where $r^l_{od}$ is 1 if there is a functional path from node $o$ to node $d$ under scenario $l$, and 0 otherwise.

To find $\sum_{o=1}^{n} \sum_{d=1}^{n} r^l_{od}$, a simple breadth-first search algorithm is used, which executes the following steps.
3.3 Solution approach

Step 1: Starting from an initial node, all adjacent nodes (connected by an edge that has not failed) are assigned to the same group. Those neighbours become the new initial nodes, and their adjacent nodes are assigned to the same group (see Figure 3.3(b)). If no more neighbours are found, go to step 2.

Step 2: The algorithm moves to the next node that has not been assigned to a group yet and applies the same steps (see Figure 3.3(c)). If no more nodes are left unassigned, the algorithm is finished (see Figure 3.3(d)), otherwise, repeat step 2.

Figure 3.3: Marking disconnected groups for calculating the connectivity index
Chapter 3 Multiple source and destination security strategy optimisation

To calculate the number of disconnected node pairs, the cardinality of each group is multiplied with that of every other group, and then summed.

Finally, to find $\text{Con}_l$, the number of disconnected node pairs, $\sum_{o=1}^{n} \sum_{d=1}^{n} r_{od}^l$, is divided with the total number of theoretical connections to get a percentage value, which is then subtracted from 1 to get the percentage of connectivity. In Figure 3.3, this would be one group of 7 and one group of 3. The total number of connections possible is 45. The rate of disconnection is 47.7%. It is important to take into account in our examples that the connection from A to B is the same as the connection from B to A, as the graph is undirected. In case of a directed graph this formula and algorithm has to be adapted to be suitable to calculate the connectivity index.

Using the probability for critical scenario $l$ to occur ($R_l$), $\rho^e_i$ and $\rho^n_k$ are defined as the potential of edge $i$ and node $k$ as follows:

\begin{align}
\rho^e_i &= \sum_l (1 - \text{Con}_l) \cdot R_l \cdot f^e_{il} \\
\rho^n_k &= \sum_l (1 - \text{Con}_l) \cdot R_l \cdot f^n_{kl}
\end{align}

(3.16)

(3.17)

where $f^e_{il}$ ($f^n_{kl}$) is 1 if the edge $i$ (node $k$) is present in set $E^F_l$ ($N^F_l$), 0 otherwise.

The potential of edge $i$ and node $k$ is an estimate of the probability of that edge or node to cause a network failure, and hence is also a
3.3 Solution approach

good measure to determine the likelihood for an upgrade of the security strategy of that edge or node to improve this probability. It uses a combination of the risk for a critical scenario to occur, weighted with the impact of this scenario (which is one minus the amount of connections left if a critical scenario takes place), and this value is then summed for all the scenarios in which that edge or node fails.

This potential is used to generate the restricted candidate list (RCL) for the GRASP heuristic, as well as, for finding promising edges and nodes for the VND. The edges and nodes are all put together in a list and sorted by their potential. The top \( \alpha \) elements will then form the RCL.

After the generation of an initial solution by the GRASP heuristic, a local search heuristic is applied during the Intensification step. The local search makes use of a VND heuristic to improve the current solution. The VND heuristic explores the neighbourhoods of three different local search operators. The first one “Upgrade” tries to upgrade one security strategy used inside an edge (or node) with another security strategy for said edge (or node). The second move, “Budget reallocation”, tries to relocate budget from one edge to another by downgrading a security strategy used for one edge (or node) and applying the budget to another edge (or node) by upgrading a security strategy on that edge (or node). The third move, “Budget redistribution”, downgrades the security strategies from two edges (or nodes) and they are then afterwards upgraded with alternative security strategies for the same two nodes or edges (this can be seen as a budget redistribution between the selected edges or nodes). A move is executed until no further improvement is
found. The whole VND heuristic is carried out until a local optimum is found and no further improvement can be obtained.

Finally, a perturbation is applied to escape this local optimum (this is a diversification step), and the algorithm continues with a local search on this perturbed solution. In this perturbation step, part of the solution is destroyed by removing security strategies from edges or nodes randomly. These security strategies are added to the tabu list for a specified number of iterations. The edges or nodes they belonged to are also added to a tabu list for a predefined number of iterations. If, after a fixed number of perturbations, the algorithm cannot find a better solution, the algorithm is restarted from a new solution constructed by the GRASP heuristic. The perturbation step in this algorithm does not require a repair step, as a solution from which a security strategy is removed is always feasible.

3.4 Computational experiments

The solution approach, described before, has been tested on a set of test instances, which are made available for download from http://antor.uantwerpen.be/downloads/NS. The instances are randomly generated, taking into account realistic features of networks and failure rates. The values of the parameters that were passed to the instance generator to generate a specific instance are encoded in the instance name.

The generated nodes that are positioned randomly in a uniform fashion are connected by a minimal spanning tree, and in a next step, edges
are randomly selected from the Delaunay triangulation of those nodes to be added to the instance. The maximum number of edges that can be added is defined in the input parameters. All the nodes and edges are assigned a probability of failure which is randomly selected, and which is of a realistic value. From Fadaee and Tabatabaei (2010), we can learn that, for a real 16 year old water network the failure rate is between 0.5% and 1.5%. Some preliminary tests reveal that the exact failure rate does not make a difference for the effectiveness and speed of the heuristic approach proposed in this chapter. Next, each edge and node are assigned between 1 and \( c \) countermeasures (\( c \) is a user-defined parameter), which are combined in security strategies for that edge or node. Each security strategy has a reduction of risk and a cost.

The encoding of the instances’ name is the same as discussed in Section 2.5.1. Forty-five test instances were generated. The first set consists of 15 instances which have five nodes each, 12 of those have five edges, three have six edges. A second set, consisting of 15 instances, has six nodes per instance of which two have six edges, and 13 have seven edges. A final set is generated, which has 15 instances which consist out of seven nodes and eight edges. For each of the generated instances, each edge and node has five available security strategies.

All computational experiments are conducted on a desktop computer running a 64 bit version of Linux. The system has a Intel® Core™ i7-4790 processor, running at 3.60GHz, and has 16GB of memory. In the worst case, for an instance with seven nodes and eight edges, \( 2^{15} \) possible critical scenario’s have to be stored in memory. Based on the data in Section 2.5.1, an estimation is made that instances with a combined
total of 26 edges and nodes are the largest instances solvable with this desktop system.

### 3.4.1 Calibration of the metaheuristic

In a first phase, the metaheuristic is tuned in order to find its best parameter settings. This tuning is done in a controlled full factorial statistical experiment on a subset of the instances. The subset entails all instances with seven nodes. The parameters that are investigated are the maximum number of iterations (\texttt{maxiter-no-improvement}), the percentage of the solution that is perturbed (\texttt{perturb-percent}), the size of the tabu list (\texttt{tabu-tenure}), the selection mechanism (\texttt{selection-mechanism}), the size of the restricted candidate list (\texttt{alpha}), and the percentage of the instance used for the third move (\texttt{double-swap-percentage}, see Section 3.3).

Analysis of variance (ANOVA) reveals that the percentage of the solution that is perturbed, the size of the tabu list, the size of the restricted candidate list, and the percentage of the instance used for the third move all have a statistically significant influence on the objective function value. A graphical output of the statistical experiment is shown in Figure 3.4. From the full factorial experiment we choose the parameter settings shown in Table 3.1. The parameter settings that resulted in the lowest objective value are selected.
3.4 Computational experiments

Figure 3.4: Plot of average objective values and computing times for given parameter setting

Table 3.1: Heuristic parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Values</th>
<th>Chosen value</th>
</tr>
</thead>
<tbody>
<tr>
<td>max-iter</td>
<td>Number of restarts</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>perturb-percent*</td>
<td>Percent of edges removed during the perturbation phase</td>
<td>10%, 30%, 50%, 70%</td>
<td>70%</td>
</tr>
<tr>
<td>alpha*</td>
<td>Size of the restricted candidate list</td>
<td>1, 2, 3, 4</td>
<td>3</td>
</tr>
<tr>
<td>tabu-tenure*</td>
<td>Number of iterations that a strategy is kept in the tabu list</td>
<td>10, 30, 50</td>
<td>30</td>
</tr>
<tr>
<td>max-iter-no-improvement</td>
<td>Number of iterations without improvements</td>
<td>5, 10, 20</td>
<td>10</td>
</tr>
<tr>
<td>double-swap-percentage*</td>
<td>Percentage value used to find the amount of evaluations to be performed by the Double swap move</td>
<td>20%, 50%, 80%</td>
<td>80%</td>
</tr>
<tr>
<td>selection-mechanism</td>
<td>Strategy to select edges</td>
<td>best (0), first(1)</td>
<td>best</td>
</tr>
</tbody>
</table>
3.4.2 Results

In a second phase, the algorithm is executed with these selected parameter settings on a larger set of instances to analyse its behaviour. The obtained results are compared with the best solutions found by the naive exact approach described in Section 3.3.2. If a solution was not found within 24 hours by the exact approach, no value is given.

In Table 3.2, the percentage of difference between the best solutions obtained by the metaheuristic over 25 runs and the exact approach is reported in column *Best Gap*, while column *Average Gap* represents the percentage gap between the average solution over 25 runs and the best solution found by the exact approach. It can be observed that the metaheuristic, in most cases, finds solutions that are equal to those found by the exact approach. Moreover, these solutions of high quality are also found in a very limited running time, i.e., a small fraction of the time used by the exact approach. An increase in computational time can be noticed between instances of five and six nodes and instances with seven nodes. The difference is caused by the increase in number of critical scenarios.

In Figure 3.5, the current and best found objective values for one instance are plotted over time. The spikes in the current solution show the points in time at which a perturbation is executed. After a perturbation, the VND heuristic lowers the objective value in small steps until it reaches a local optimum, after which another perturbation is executed. The objective function has a very steep descent at the start of the algorithm and gradually stabilizes. This shows that the algorithm
### 3.4 Computational experiments

Table 3.2: Results of the exact approach compared with those of the metaheuristic

<table>
<thead>
<tr>
<th>Instance</th>
<th>Exact Approach</th>
<th>Metaheuristic</th>
<th>Metaheuristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>NS-n5-c3-C3-a30-x0</td>
<td>0.064</td>
<td>0.064</td>
<td>0.080</td>
</tr>
<tr>
<td>NS-n5-c3-C3-a30-x1</td>
<td>0.051</td>
<td>0.051</td>
<td>0.069</td>
</tr>
<tr>
<td>NS-n5-c3-C3-a30-x10</td>
<td>0.057</td>
<td>0.057</td>
<td>0.066</td>
</tr>
<tr>
<td>NS-n5-c3-C3-a30-x11</td>
<td>0.064</td>
<td>0.064</td>
<td>0.058</td>
</tr>
<tr>
<td>NS-n5-c3-C3-a30-x12</td>
<td>0.060</td>
<td>0.060</td>
<td>0.067</td>
</tr>
<tr>
<td>NS-n5-c3-C3-a30-x13</td>
<td>0.065</td>
<td>0.065</td>
<td>0.051</td>
</tr>
<tr>
<td>NS-n5-c3-C3-a30-x14</td>
<td>0.059</td>
<td>0.060</td>
<td>0.062</td>
</tr>
<tr>
<td>NS-n5-c3-C3-a30-x2</td>
<td>0.046</td>
<td>0.046</td>
<td>0.048</td>
</tr>
<tr>
<td>NS-n5-c3-C3-a30-x3</td>
<td>0.062</td>
<td>0.062</td>
<td>0.067</td>
</tr>
<tr>
<td>NS-n5-c3-C3-a30-x4</td>
<td>0.062</td>
<td>0.062</td>
<td>0.069</td>
</tr>
<tr>
<td>NS-n5-c5-C3-a30-x5</td>
<td>0.055</td>
<td>0.055</td>
<td>0.057</td>
</tr>
<tr>
<td>NS-n5-c5-C3-a30-x6</td>
<td>0.045</td>
<td>0.045</td>
<td>0.046</td>
</tr>
<tr>
<td>NS-n5-c5-C3-a30-x7</td>
<td>0.063</td>
<td>0.063</td>
<td>0.069</td>
</tr>
<tr>
<td>NS-n5-c5-C3-a30-x8</td>
<td>0.051</td>
<td>0.051</td>
<td>0.056</td>
</tr>
<tr>
<td>NS-n5-c5-C3-a30-x9</td>
<td>0.056</td>
<td>0.056</td>
<td>0.060</td>
</tr>
<tr>
<td>NS-n6-c5-C3-a30-x10</td>
<td>0.062</td>
<td>0.062</td>
<td>0.063</td>
</tr>
<tr>
<td>NS-n6-c5-C3-a30-x11</td>
<td>0.054</td>
<td>0.054</td>
<td>0.056</td>
</tr>
<tr>
<td>NS-n6-c5-C3-a30-x12</td>
<td>0.056</td>
<td>0.056</td>
<td>0.170</td>
</tr>
<tr>
<td>NS-n6-c5-C3-a30-x13</td>
<td>0.074</td>
<td>0.074</td>
<td>0.186</td>
</tr>
<tr>
<td>NS-n6-c5-C3-a30-x14</td>
<td>0.082</td>
<td>0.082</td>
<td>0.155</td>
</tr>
<tr>
<td>NS-n6-c5-C3-a30-x2</td>
<td>0.067</td>
<td>0.067</td>
<td>0.200</td>
</tr>
<tr>
<td>NS-n6-c5-C3-a30-x3</td>
<td>0.066</td>
<td>0.066</td>
<td>0.250</td>
</tr>
<tr>
<td>NS-n6-c5-C3-a30-x4</td>
<td>0.062</td>
<td>0.062</td>
<td>0.165</td>
</tr>
<tr>
<td>NS-n6-c5-C3-a30-x5</td>
<td>0.076</td>
<td>0.076</td>
<td>0.112</td>
</tr>
<tr>
<td>NS-n6-c5-C3-a30-x6</td>
<td>0.063</td>
<td>0.063</td>
<td>0.169</td>
</tr>
<tr>
<td>NS-n6-c5-C3-a30-x7</td>
<td>0.064</td>
<td>0.064</td>
<td>0.300</td>
</tr>
<tr>
<td>NS-n6-c5-C3-a30-x8</td>
<td>0.063</td>
<td>0.063</td>
<td>0.216</td>
</tr>
<tr>
<td>NS-n7-c5-C3-a30-x10</td>
<td>0.097</td>
<td>0.097</td>
<td>5.069</td>
</tr>
<tr>
<td>NS-n7-c5-C3-a30-x11</td>
<td>0.075</td>
<td>0.075</td>
<td>5.082</td>
</tr>
<tr>
<td>NS-n7-c5-C3-a30-x12</td>
<td>0.083</td>
<td>0.083</td>
<td>2.856</td>
</tr>
<tr>
<td>NS-n7-c5-C3-a30-x13</td>
<td>0.050</td>
<td>0.050</td>
<td>3.286</td>
</tr>
<tr>
<td>NS-n7-c5-C3-a30-x14</td>
<td>0.081</td>
<td>0.081</td>
<td>3.453</td>
</tr>
<tr>
<td>NS-n7-c5-C3-a30-x2</td>
<td>0.068</td>
<td>0.068</td>
<td>4.550</td>
</tr>
<tr>
<td>NS-n7-c5-C3-a30-x3</td>
<td>0.085</td>
<td>0.085</td>
<td>4.225</td>
</tr>
<tr>
<td>NS-n7-c5-C3-a30-x4</td>
<td>0.097</td>
<td>0.097</td>
<td>6.083</td>
</tr>
<tr>
<td>NS-n7-c5-C3-a30-x5</td>
<td>0.084</td>
<td>0.084</td>
<td>4.897</td>
</tr>
<tr>
<td>NS-n7-c5-C3-a30-x6</td>
<td>0.085</td>
<td>0.085</td>
<td>3.049</td>
</tr>
<tr>
<td>NS-n7-c5-C3-a30-x7</td>
<td>0.091</td>
<td>0.091</td>
<td>2.690</td>
</tr>
<tr>
<td>NS-n7-c5-C3-a30-x8</td>
<td>0.083</td>
<td>0.083</td>
<td>4.173</td>
</tr>
<tr>
<td>NS-n7-c5-C3-a30-x9</td>
<td>0.078</td>
<td>0.078</td>
<td>4.591</td>
</tr>
</tbody>
</table>
converges towards very good results in a very short time. As CPU time gets larger, the marginal improvement of the best solution found so far becomes smaller and smaller. This shows that infinitely increasing the CPU time will not necessarily produce significantly better results.

### 3.5 Conclusion and discussion

In this chapter, a model for the selection of appropriate security strategies given a limited budget is described. Its goal is to increase the security of infrastructure such as pipelines transportation systems, telecommunication networks, smart grids, etc.

Chapter 2 was extended by adding the assumption that nodes have a probability of failure and designated security strategies too. Furthermore, each node was a potential source and destination, which required
3.5 Conclusion and discussion

an adaptation for the calculation of the objective value. Also, the definition of a disconnection had to be redefined.

Very good results are obtained by the proposed heuristic on the provided instances. In most cases, it finds the optimal solution in a fraction of the time that is needed by the exact approach. The parameters that have a significant influence on the results are the perturbation percentage, where a higher percentage yielded better results, the tabu tenure, the number of elements in the restricted candidate list, alpha, and the percentage of edges and nodes on which the budget redistribution move is executed.

However, due to the fact that we need to generate and evaluate all scenarios to find the critical scenarios and the sheer number of critical scenarios, to tackle larger instances, a different approach should be investigated. An approach which uses minimal cut sets, as discussed in Section 2.6, could be considered. However, this approach should get adapted to support the multiple source and destination nodes which are used in the model described in this chapter.

In future research, the model can be extended even more by making a differentiation in the types of nodes (customers or suppliers) and/or considering the importance of nodes themselves. In terms of the algorithm, alternative solution approaches can also be developed and compared with the one proposed in this chapter. A very good implementation of an exact approach should be considered, to find the optimal solution for a set of test instances, in a reasonable amount of time.
The mitigation of domino effects through the allocation of protective safety barriers

This chapter is based on the manuscript “Janssens, J., Talarico, L., Reniers, G. L., and Sörensen, K. (2015a). A decision model to allocate protective safety barriers and mitigate domino effects. Reliability Engineering & System Safety, 143:44 – 52”
4.1 Introduction

Cascade events or domino effects truly are a timely topic. Domino effects can be defined as accidents in which a primary unwanted event propagates within a system (“temporally”), or/and to nearby systems (“spatially”), sequentially or simultaneously, triggering one or more secondary unwanted events, in turn possibly triggering further (higher order) unwanted events, resulting in overall consequences more severe than those of the primary event (Cozzani and Reniers, 2013). In this chapter, cascade events and domino effects are treated as synonyms, even though, the former are mainly used in works related to social and organizational effects/accidents, while the latter are generally mentioned in technical studies. We live in a time where industrial activities are even more widespread, especially the chemical and process industry. This translates into a non-stop increase in amounts of hazardous materials being processed, stored and transported in and between chemical plants in industrial parks worldwide. The need for more industrial activity is driven by the observation that population figures have been sharply increasing on a global scale since a century. Irrespective of the underlying reasons of both facts, taking the combination of both these facts into consideration, automatically leads to the question about their combined impact on societal risk and safety. In the chemical industry, an important aspect of this impact can be summarized by the potential of escalation of an industrial accident to a major disaster, or a so-called domino effect.

Although such events are less known than well-recognized major accidents such as for example vapour cloud explosions (VCEs), boiling
Chapter 4 Mitigating domino effects by allocating protective safety barriers

Liquid expanding vapour explosions (BLEVEs), and the like, they may have even more disastrous consequences compared to those better known accidents (Khan and Abbasi, 2001). They are less recognized and studied by industry, academia and regulators due to the fact that their likelihood is even much lower than that of the better known major accident scenarios. Nonetheless, since they became an issue in the Seveso II Directive (Council of European Union, 1996), and also because domino accidents do happen all over the world (even if they are extremely rare), ever more research is necessary and carried out by academics and industry to further advance our knowledge on these obscure events.

Several lines of research have been initiated with respect to the domino effect topic. For example, indices have been suggested by Tugnoli et al. (2008) and Reniers and Audenaert (2014). Tugnoli et al. (2008) developed an index to assess the domino potential hazard including the effect of inherent and passive protection measures. Reniers and Audenaert (2014) elaborated an index to rank chemical installations within any industrial area based on their vulnerability for domino effects. Nguyen et al. (2009) analysed the potential for domino effects produced by projectiles generated by explosions in industrial facilities. Salzano et al. (2014) investigated domino effects related to home-made explosives. Landucci et al. (2009) elaborated a quantitative risk assessment where domino effects are taken into account, and where events are triggered by fire. The model is based on an estimation of vessel time to failure. Cozzani et al. (2009) studied inherent safety approaches providing the possibility to prevent knock-on events. Khakzad et al. (2013) proposed an approach to analyse domino effects by using Bayesian networks. Reniers (2010) looked into the problem of cross-plant col-
laboration and the lack of sufficient information exchange to optimise protection against domino effects, employing game-theoretical modelling to do so. Darbra et al. (2010) analysed 225 domino incidents during hazmat transportation. Reniers et al. (2014) investigated the possibility of attenuation-based security within chemical industrial areas. Furthermore, in 2013, Reniers and Cozzani (2013a) edited a comprehensive volume on the modelling, prevention and management of domino effects in the process industries, providing the state-of-the-art at publication date and indicating the leeway for further exploration of the domino effects research area. As can be seen from this brief overview of important past research on domino effects, the subject is looked at from a safety as well as from a security point of view, and research efforts are ever more intensifying.

A lot of research is concerned with design-based safety with respect to domino effects, and hence, researchers mainly focus on managing domino effects in an inherent manner. This is, of course, the most optimal way to deal with such potentially devastating events. However, this is not always possible or sufficient. If installations (for example storage tanks) are present in a certain industrial setting, it is not easy to just replace them or to make major design-based (e.g., lay-out) changes. Therefore, it is also critical that research is aimed at optimising add-on safety with respect to domino effects. The study explained and discussed in this chapter is aimed at such optimisation of safety barriers within existing industrial settings, and employs operational research techniques and science to do so.

The concept of barrier is widely used to denote some form of obstruc-
tion towards an emerging threat or accident (Lindoee and Stene, 2011). Different terms (barrier, defence, protection layer, safety critical elements, safety function, etc.) are used in the literature to describe barriers as risk reducing measures (the reader is referred to Sevcik and Gudmestad (2014) for more details). Even though there does not exist neither a universally accepted definition of safety barriers nor any agreement regarding their effects, some common features (e.g., barrier systems, barrier functions, safety elements) can be found in the literature (Sklet, 2006). In order to overcome this issue, the Norwegian Petroleum Safety Authority outlined specific definitions for safety barriers, safety functions, safety elements. In particular barriers are defined as “...technical, operational and organisational elements on an offshore or onshore facility, that, individually or collectively, reduce the possibility of concrete failures, hazard and accident situations occurring, or that limit or prevent harm/inconveniences”. Moreover, barriers are intended either to prevent a concrete chain of events from occurring or to affect a chain of events in a way that limits harm and/or losses. Barriers fulfil their functions in case of failures, hazard and accident situations on an offshore or onshore facility, be it a case of potential harm done to people, the external environment and/or financial assets (Petroleum Safety Authority Norway, 2014). In the remainder of this chapter, we will refer to safety barriers having in mind the concepts provided by the Norwegian Petroleum Safety Authority.

The evolution of domino accidents, triggered by heat radiation, overpressure effects, or missile projection, depends on the presence (or absence) and the performance of safety barriers. Safety barriers may have the potential to prevent escalation, for example, in case of heat
4.1 Introduction

Radiation, delaying or avoiding the heat-up of secondary targets. Thus, safety barriers play a crucial role in domino effect prevention and mitigation within existing industrial settings. More specifically, add-on safety barriers can indeed: (i) restrict the propagation of domino effects; (ii) mitigate the consequences of domino effect; and (iii) be extremely important in terms of increasing the time to failure of chemical installations.

Today, in industrial practice, the decision to take certain safety barriers for dealing with major accident scenarios does not take domino effects of a higher order into account. At most, possible direct escalation of major accident scenarios is considered (thus only possible domino events with cardinality 0, see Section 4.2). However, this is a myopic way of tackling domino effects within chemical plants. Especially with respect to safety issues, this myopic approach may prove to be largely insufficient. Therefore, to optimise current practice, there is need for research to investigate in what way higher order domino events can be taken into account in the decision-making process of how and where to invest in add-on safety barriers for existing industrial areas. Possibly, considering higher-order domino events in the safety barrier investment problem will lead to alternative decisions. An approach and a computer program to determine the most optimal safety barrier investment decision for dealing with domino effects in existing industrial settings, and thereby considering higher-order domino events, is currently non-existent in academic literature and lacking in industrial practice.

The remainder of the chapter is organized as follows. In Section 4.2,
the decisional model and its mathematical representation is presented. In Section 4.3, an effective solution algorithm based on a metaheuristic approach is developed. This solution method is tuned and tested on a realistic case study in Section 4.4. Section 4.5 concludes the chapter and presents some suggestions for future research.

### 4.2 Problem description

In this section, the problem is described and mathematically stated. The main objective of the model is to support decision makers to optimally locate protective barriers within an industrial setting of chemical installations, to mitigate domino effects. Given a budget constraint, the optimal mix of protective barriers needs to be selected in order to delay the propagation of a major fire resulting from an accident at a chemical installation that might further trigger the failure of other chemical installations, thus engendering escalation effects.

Depending on the intensity of the domino effects, the cardinality $D$ can be used to denote how many domino events after the initiating failure/accident are considered by the decision maker. We suppose that the initiating event always happens at a root installation and from it, fire might propagate to neighbouring installations starting a cascade effect.

Domino events characterized with cardinality 0 represent the first cascade effect as a consequence, e.g., of an accident to a chemical installation (the so-called “primary domino events”), where as cardinality 1
4.2 Problem description

Figure 4.1: Situation I and Situation II in terms of cardinality

refers to secondary domino events, cardinality 2 to tertiary domino events and so on (Reniers and Cozzani, 2013b). It is worth noticing that, when the cardinality is equal to zero the first *domino* effect is taking place. Using this taxonomy, it is possible to classify domino effects triggered by installation \( i \) and affecting:

(i) **Situation I**: a single neighbour installation \( j \) by means of fire propagating from \( i \) to \( j \) (in case \( D = 0 \));

(ii) **Situation II**: a neighbour installation \( j \) and an installation \( l \), that is a neighbour of \( j \), by means of fire propagating from \( i \) to \( j \) and subsequently from \( j \) to \( l \) (in case \( D = 1 \)) and so on.

In Figure 4.1, both situations are shown. In the remainder of the chapter, for the sake of clarity of exposition when we represent the initiating event resulting in the first domino event of cardinality 0 affecting the root node \( i \), we implicitly assume that a major accident (e.g., a major fire) has already affected installation \( i \).

The domino effects analysed in this chapter are mainly related to the so called “point-source” scenarios where an initiating accident, affecting
an installation, propagates within a chemical system by means of fire or heat radiation. In these cases, the cardinality of the domino effects together with the physical location of the neighbouring installations play a significant role in the allocation of safety barriers. For other types of domino effects related e.g., to fragment/debris, explosion and flash fire, the propagation of the triggering (initial) scenario and the consequences of the domino accident might depend on the geometry of the system (the cloud - concentration and shock wave) rather than cardinality of the domino effects. Further studies, aimed at extending the proposed model to consider the specific features of these type of domino accidents, are left for future work.

In this chapter, an industrial area that is potentially subjected to domino effects, such as a chemical plant, is modelled by using a graph, $\mathcal{G} = \{N, \mathcal{E}\}$. $N$ is the set of nodes representing the critical installation within the industrial area that, after an accident, may engender domino effects. $\mathcal{E}$ denotes the set of edges $(i, j)$ representing a fire propagation from node $i$ to node $j$. As a consequence of a failure/accident happening to node $i \in N$ fire propagates, along edge $(i, j) \in \mathcal{E}$, in a non negative propagation time $pt_{ij}$, triggering a failure/accident of a neighbouring node $j \in N$. In other words, $pt_{ij}$ represents the time needed by the fire to propagate from node $i$ to $j$ and to cause a failure of $j$ without any safety measure taken.

A set, $M_{ij}$, is defined for each edge $(i, j)$ and comprises all the protective measures that are available for this edge. Each protective measure $k$ for edge $(i, j)$ presents a cost $c_{ij}^k$ and a value of effectiveness $e_{ij}^k$ in delaying the escalation and thus increasing the propagation time needed by
4.2 Problem description

fire to affect a neighbour facility \( j \) starting from node \( i \). Both cost and effectiveness associated to each protective measure are based on information, such as number and type of protective barriers, thickness, equipments and used materials. These values are assumed to be predefined by a safety risk assessment, carried out by the safety management team.

Let \( B \) represent the maximum available budget to be invested in protective measures. For the sake of simplicity, for each edge \((i, j)\), a dummy protective measure, having a cost \( c_{ij}^0 = 0 \) and an effectiveness \( e_{ij}^0 = 0 \), is defined. It represents a default state that indicates that no protective measure is applied on edge \((i, j)\). Moreover, only one protective measure per edge can be applied. A protective measure can be a combination of single protective barriers presenting different capabilities, to stop or delay the fire propagation, depending on the characteristics of the barriers themselves. A combination of protective barriers can have a different effectiveness (greater or lower, e.g., in case of an increased complexity needed to activate the combined barriers in case of accident or in case of materials and/or construction constraints which limit the performance of the combined measure) than the sum of the impact of the individual protective barrier due to possible interaction effects. In some cases, specific combinations of single barriers might not be available due to possible incompatibility factors (see e.g., Table 4.1).

In order to make the notation used inside the mathematical model more readable, a set, \( F_i^D \) for each node \( i \in \mathcal{N} \), is defined. Set \( F_i^D = \{P_1, P_2, \ldots, P_q\} \) contains a list of \( q \) fire-paths denoting all possible cascading effects of cardinality \( D \) that can be triggered by a fail-
Table 4.1: Set of protective measures $\mathcal{M}_{ij}$ for edge $(i,j)$

<table>
<thead>
<tr>
<th>Measure id</th>
<th>Combination of protective barriers</th>
<th>Cost</th>
<th>Effectiveness</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>100</td>
<td>0.15</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>150</td>
<td>0.25</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td>200</td>
<td>0.4</td>
</tr>
<tr>
<td>4</td>
<td>A&amp;B</td>
<td>250</td>
<td>0.45</td>
</tr>
<tr>
<td>5</td>
<td>B&amp;C</td>
<td>300</td>
<td>0.66</td>
</tr>
</tbody>
</table>

The generic fire-path $P_k \in \mathcal{F}_i^D$ is composed by a sequence of $D+1$ edges starting from root node $i$ (e.g., $(i,j), (j,l), (l,m), \ldots$) resulting in an escalation (i.e. accident/failure) that affects a sequence of $D + 2$ nodes of the graph $G$ (i.e. nodes $i$ and $j$ in case of $D = 0$).

It is worth noting that the cardinality of the domino accidents $D$, which is used inside the model as a input parameter, should be defined by a risk expert during a preliminary hazard identification phase. The model, proposed in this chapter, could also be used by a risk analyst during the hazard identification phase as a decision support tool, simulating and assessing the impact of different values associated to $D$.

To mathematically state the problem, three families of decision variables are defined: (1) Let $PT_{ij}$ be the propagation time of the fire along edge $(i,j)$ when at least one protective measure is used; (2) Let $ET_i$ be the escalation time after which a domino effect of cardinality $D$ is initiated as a consequence of a failure/accident happened at node $i$; Let $x_k^{ij}$
be a binary decision variable that is equal to one if protective measure 
k for edge \((i, j)\) is selected, zero otherwise. The decision problem is
defined as follows:

\[
\text{lex max } f(x) = (f_1(x), f_2(x)) \tag{4.1}
\]

\[
f_1(x) = \min_{i \in \mathcal{N}} ET_i \tag{4.2}
\]

\[
f_2(x) = \sum_{(i, j) \in \mathcal{P}_i} PT_{ij} \quad \forall P_i \in \mathcal{F}_i^D, \forall i \in \mathcal{N} \tag{4.3}
\]

s.t.

\[
\sum_{(i, j) \in \mathcal{E}} \sum_{k \in \mathcal{M}_{ij}} c_{ij}^k \cdot x_{ij}^k \leq B \tag{4.4}
\]

\[
PT_{ij} = \sum_{k \in \mathcal{M}_{ij}} pt_{ij} \cdot (1 + e_{ij}^k) \cdot x_{ij}^k \quad \forall (i, j) \in \mathcal{E} \tag{4.5}
\]

\[
\sum_{k \in \mathcal{M}_{ij}} x_{ij}^k = 1 \quad \forall (i, j) \in \mathcal{E} \tag{4.6}
\]

\[
ET_i \leq \sum_{(i, j) \in \mathcal{P}_i} PT_{ij} \quad \forall P_i \in \mathcal{F}_i^D, \forall i \in \mathcal{N} \tag{4.7}
\]

\[
x_{ij}^k \in \{0, 1\} \quad \forall (i, j) \in \mathcal{E}, \forall k \in \mathcal{M}_{ij} \tag{4.8}
\]

The objective function \(f(x)\) in Equation (4.1) is used to evaluate the
quality of feasible solutions. It is divided into two objectives \(f_1(x),
\)
\(f_2(x)\), to be both maximised in a lexicographic order. The lexicographic
ordering assumes that the objectives \(f_1(x)\) and \(f_2(x)\) can be ranked by
the decision maker in order of importance. In this chapter we assume,
without loss of generality, that the objective functions \(f_1(x)\) and \(f_2(x)\)
are in the order of importance so that \(f_1(x)\) is the most important and
More specifically, the objective function $f(x)$ maximise in order:

(i) function $f_1(x)$ (in Equation (4.2)) namely the escalation time associated to the worst case scenario presenting the lowest total escalation time considering a domino effect of cardinality $D$ in which a sequence of $D$ node accidents in cascade are triggered by a failure at root node $i$; (ii) function $f_2(x)$ (in Equation (4.3)) namely the sum of the propagation time associated to all possible scenarios with an accident in any of the nodes triggering a domino effect of cardinality $D$. This objective attempts to increase the effectiveness of the safety barriers considering not only the worst case scenario, but taking into account the mitigations of possible accidents affecting the overall industrial area. The ranking of solutions is based on a multi-objective lexicographic order, i.e. a solution $x$ is considered better than $x'$ if and only if, $f_1(x) > f_1(x')$ or $f_1(x) = f_1(x')$ and $f_2(x) > f_2(x')$. In other words, $x$ is preferred to $x'$ if and only if, for some $i \in [1, 2]$, $f_i(x) > f_i(x')$ and for all $j$ such that $j < i$, $j \geq 1$, $f_j(x) = f_j(x')$.

In other words, a solution with a higher value of $f_1(x)$ is always preferred. In case of solutions with an equal value of $f_1(x)$, the one with the highest value of $f_2(x)$ is to be selected. Constraint (4.4) guarantees that the total cost associated to the selected protective barriers does not exceed a predefined budget $B$. Constraint (4.5) is used to define the propagation time $PT_{ij}$ associated to edge $(i, j)$ depending on the type of protective measures being installed on that edge. Constraint (4.6) forces the decision process to select at maximum one protective meas-
ure to increase the propagation time associated to edge \((i, j)\). It should be noted that \(x_{ij}^0 = 1\) means that for edge \((i, j)\) no protective system of barriers has been applied. Constraint (4.7) is used to compute for each node the minimum escalation time given an accident that happens in node \(i\) and generates a domino effect with a cardinality \(D\). Finally, Constraint (4.8) represents the domain of the decision variables, which ensures that no partial protective measures are allowed.

4.3 Solution approach

The problem described in Equation (4.1)-(4.8), belongs to the class of knapsack problems, also known as resource allocation problems. These well-known combinatorial optimisation problems have been widely studied in the literature (see e.g., Wilbaut et al. (2008)). In general, knapsack problems include a set of items each with a certain benefit and cost. The goal is to select a subset of these items in order to maximise the benefit within a certain budget.

As problem instances grow larger, an exact algorithm will require an exponential amount of time to solve them. Therefore, the optimality is sacrificed for near optimal solutions, that can be calculated in a very short amount of time. To achieve this goal, metaheuristics are used. The solution approach developed in this chapter is based on a tabu search heuristic hybridized with an iterated local search that makes use of a variable neighbourhood descent heuristic (Hansen and Mladenović, 2001). The overall structure of the metaheuristic is shown in Algorithm 4.1.
Algorithm 4.1 Pseudo-code of the metaheuristic solution approach

1: **Phase 1: Pre-computation**
2: Define the list of paths of length $D$, for each node
3: Compute the escalation time $ET$ for each path
4: **Phase 2: Generation of initial solution**
5: let $x$ be the current solution and $f(x)$ its cost
6: let $x^*$ be the best solution found so far and $f(x^*)$ its cost
7: $x^*, x \leftarrow \emptyset$, $f(x^*), f(x) \leftarrow \infty$
8: Initialize the tabu list $TB$, initially empty
9: $x \leftarrow$ GRASP Heuristic($TB$)

10: **Phase 3: Intensification stage**
11: while (stopping criterion not reached) do
12: $k \leftarrow 0$
13: while ($k < 3$) do
14: $x' \leftarrow N_k(x)$
15: if $((f_1(x) < f_1(x'))$ or $f_1(x) = f_1(x')$ and $f_2(x) < f_2(x'))$
16: then
17: $x \leftarrow x'$
18: else
19: increase $k$
20: end if
21: end while
22: if $((f_1(x^*) < f_1(x))$ or $f_1(x^*) = f_1(x)$ and $f_2(x^*) < f_2(x))$ then
23: $x^* \leftarrow x, f(x^*) \leftarrow f(x)$
24: end if
25: update number of iterations without improvement

26: **Phase 4: Diversification stage**
27: if (max number of iterations without improvement not reached) then
28: $x \leftarrow$ Perturbation($x$)
29: update $TB$
30: else
31: $x \leftarrow$ GRASP heuristic($TB$)
32: end if
33: end while
34: return $x^*$
4.3 Solution approach

In *phase 1*, some preliminary computations are made in order to speed up the solution process. In particular, given the cardinality of domino scenarios that need to be considered, the list of paths, through which the accident originating in a given node can propagate, is generated. In other words, all the *fire-paths* $P_k \in \mathcal{F}_i^D$ having as origin node $i$ and cardinality $D$ are generated $\forall i \in \mathcal{N}$. Moreover, for each of these paths the total escalation time without any protective measures is computed using the following formula: $\sum_{(l,h) \in P_k} p_{lh}$.

In *phase 2*, an initial solution for the problem is constructed step by step by using a GRASP heuristic (Feo et al., 1991). This method selects, in a greedy randomised fashion, one protective measure at each iteration until there is no more budget available. The selected measure must not be contained in the tabu list, $TB$, and there should be enough available budget to include it in the current solution. The paths are ordered by increasing total escalation time and then the first $\alpha$ edges for which no protective measure is applied yet, are inserted in a restricted list $RL$. Next, an edge is randomly selected from this list and a protective measure, not in the tabu list and whose cost is lower than the remaining budget, is randomly selected and added to the current solution.

*Phase 3*, improves the current solution by means of a variable neighbourhood descent. The different neighbourhoods make use of a restricted candidate list, in which edges are sorted based on their priority (based on the priority of the fire-paths they belong to, and their respective time-to-failure). For each of the neighbourhoods a best improvement strategy is applied. Three different neighbourhoods are defined as follows:(1) *Internal Swap* ($N_1$) replaces a protective measure for a given
edge with another one, that is not contained in the tabu list, for which there is sufficient budget; (2) External Swap \((N_2)\) substitutes a protective measure of one edge with another one, associated to a different edge that is not yet considered in the current solution. The substitution must be compatible with the budget constraint and allowed by the tabu list; (3) Double Swap \((N_3)\), which is a variant of the Internal Swap, executes two moves simultaneously. Two edges are selected and their protective measures removed from the current solution. The budget made available is summed with the remaining budget and used to add two new protective measures to those selected edges.

A diversification mechanism is implemented in phase 4 to let the metaheuristic escape from local optima and to explore different areas of the search space. If a maximum number of iterations without having improved the best known solution is not reached, a perturbation heuristic is applied, otherwise a new solution is built from scratch by re-applying the GRASP heuristic, which is described before. The perturbation partially removes a certain amount of protective measures from the current solution and adds them in the tabu list. If the remaining budget, after the removal operations, allows the introduction of new unexplored protective measures, they are added into the new current solution in a greedy random fashion as done in the GRASP heuristic.

As in the iterated local search framework (see Lourenço et al. (2010)), phases 3 and 4 of the proposed metaheuristic are repeated until a stopping criterion is met. This criterion needs to be defined by the user and it is usually expressed either as a maximum number of repetitions or, alternatively, as a maximum allowed computation time.
4.4 Case study

The metaheuristic, described in Section 4.3, has been programmed in the C++ language, in order to conduct the experiments introduced in this chapter. After having tuned the solution approach, an illustrative case study has been solved. In Section 4.4.1, the characteristics of the problem instance are described, while Section 4.4.2 reports the main results of the experimental analysis. A machine with an Intel® Core™ i7-2760QM processor, clocked at 2.40GHz, and 8GB RAM has been used to run the tests.

4.4.1 Test instance

Both the decision model and its related solution approach are tested on a case study instance representing a storage park of a chemical company. We chose this industrial setting because it is a realistic representation of an actual chemical park concerned with potential cascade effects. Moreover, since we use this case study for illustrative purposes, to show the reader how our metaheuristic can be applied to the domino effect research problem, we kept the example as simple as needed.

To be specific, this case study is an industrial park composed of 11 storage tanks with different characteristics such as floating roof or not, differing type of material, different sizes and variable chemical substances. For illustrative reasons, we show a storage park and its schematic representation by using a network provided in Figure 4.2. The figure depicts a network scheme overlaying an image retrieved from Google
Figure 4.2: Network scheme overlaying an image retrieved from Google maps showing a park of chemical storage tanks

maps, showing a park of chemical storage tanks. For the sake of clarity bidirectional edges (i.e. (i, j) and (j, i)) are represented by a segment between (i, j).

More specifically, a graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is used to model the storage park where $\mathcal{N}$ is the set of facility nodes (i.e. the storage tanks) and $\mathcal{E}$ is the set of edges representing possible propagation links in case of accidents to a storage tank. For example, an accident to node $i$ may trigger an accident to the neighbour facility $j$ by means of the propagation of fire from $i$ to $j$ along edge $(i, j)$.

The value associated to each edge $(i, j) \in \mathcal{E}$ represents the time needed by the fire originated in node $i$ to reach facility $j$ and to cause facility $j$ to fail. The case study instance can be downloaded from http://antor.uantwerpen.be/downloads/NS. The time for the fire to propagate from an installation to another one has been supposed proportional
to the distance between nodes including also the impact of the average weather condition such as wind. The failure times associated with the installations in \( N \) are summarized in Table 4.2. The time to failure associated to a node can be expressed as the minimum time for the fire to get uncontrollable within the node, and to start spreading in the installation. These times are related both to the characteristics of the tanks and to real industrial information concerning the exposure of tanks to atmospheric conditions.

The values displayed in Table 4.2 and Table 4.3, which are used to simulate an illustrative scenario in case of a domino accident, were validated by the head of the fire fighter department of a major chemical company. Therefore, the metaheuristic exercise on our illustrative case study instance can be considered to be realistic.

### Table 4.2: Time to failure associated to the nodes

<table>
<thead>
<tr>
<th>Facility</th>
<th>Description</th>
<th>Time to failure (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-1-3-4</td>
<td>Small tanks without any protection (Diameter ( \leq 25 ) meter)</td>
<td>20</td>
</tr>
<tr>
<td>2-5</td>
<td>Large tanks (Diameter ( \geq 30 ) meter)</td>
<td>35</td>
</tr>
<tr>
<td>6-7-8-9-10</td>
<td>Small tanks with protection (Diameter ( \leq 25 ) meter)</td>
<td>25</td>
</tr>
</tbody>
</table>

For each edge, we considered a list of protective safety barriers that can be implemented to stall the fire spread. Each barrier is characterised by a capacity to delay the propagation of fire, as well as a cost, as shown in Table 4.3. A maximum budget has been considered in the remainder of the chapter, which is set equal to EUR 3.5 Million.
Chapter 4 Mitigating domino effects by allocating protective safety barriers

Table 4.3: List of protective safety barriers

<table>
<thead>
<tr>
<th>Id</th>
<th>Barrier</th>
<th>Cost (1000 EUR)</th>
<th>Effectiveness (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No Barrier</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>Automatic sprinkler installation with additional foam</td>
<td>350</td>
<td>75</td>
</tr>
<tr>
<td>2</td>
<td>Automatic sprinkler installation without additional foam</td>
<td>250</td>
<td>65</td>
</tr>
<tr>
<td>3</td>
<td>Deluge system (water spray system opened as signalled by a fire alarm system)</td>
<td>200</td>
<td>50</td>
</tr>
<tr>
<td>4</td>
<td>Fire-resistant coating</td>
<td>180</td>
<td>45</td>
</tr>
<tr>
<td>5</td>
<td>Concrete wall surrounding tank + sprinkler without additional foam</td>
<td>2250</td>
<td>100</td>
</tr>
</tbody>
</table>

4.4.2 Results

Based on the realistic test case, discussed before, we tested the metaheuristic described in Section 4.3. The goal is to shown that it is a flexible and effective decision support tool, especially when domino effects need to be considered. Some pilot experiments have been run to tune the metaheuristic. After these preliminary test studies, the internal parameters of the metaheuristic have been set to the values reported in Table 4.4.

The metaheuristic converges towards stable solutions in a relative small number of iterations. The time needed to solve the instance slightly increases with the cardinality of the domino effects that the user wants to analyse and in the worst case takes less than 1 second.

We solved the case study, testing different values of cardinality for the domino effects. As expected, the allocation of the protective safety
4.4 Case study

Table 4.4: Metaheuristic parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repetition</td>
<td>Number of times the whole metaheuristic is repeated</td>
<td>50</td>
</tr>
<tr>
<td>IterNoImprovement</td>
<td>Maximum number of iterations without improvements</td>
<td>10</td>
</tr>
<tr>
<td>Alpha</td>
<td>Size of the restricted candidate list in the GRASP heuristic</td>
<td>5</td>
</tr>
<tr>
<td>TabuTenure</td>
<td>Number of iterations that a barrier is kept in the tabu list</td>
<td>30</td>
</tr>
<tr>
<td>Perturbation</td>
<td>Percentage number of barriers to be removed from the current solution during the perturbation phase</td>
<td>10%</td>
</tr>
</tbody>
</table>

barriers differs while domino effects that have different cardinalities are taken into account. In a myopic optimisation approach, in which there is only one domino event, as shown in Situation I described in Figure 4.1(a), (i.e. the case in which the cardinality of the domino effects is set to 0), the goal is to allocate safety barriers to stop the escalation and thus the rise of secondary (tertiary, ...) accidents triggered by a failure of an installation. However, in reality, domino effects cannot always be prevented from happening, or human intervention cannot always happen in time to prevent further escalation. Therefore, a specific allocation of protective safety barriers may be more suitable to stall the escalation and mitigate the consequence of cascade effects within a plant. A protective barrier might be positioned very well to stop the primary escalation event, but if the time needed for human intervention is longer than the time given by the protective barrier, and no further barriers are present in the system, this will lead to a catastrophe.
Taking secondary or tertiary domino effects into account when placing protective barriers, might prevent this from happening.

In a planning phase, when the design of an industrial area should be defined to cope with domino effects, several scenarios can be tested by the decision maker. Optimised allocations of safety barriers for each domino scenario can be evaluated in order to increase the time needed by the domino accident of a given cardinality to propagate through the plant. In Figure 4.3 several allocations of the available safety barriers are proposed for different values of the domino cardinality. The thickness of each edge is proportional to the effectiveness of safety barriers allocated on that edge.

Despite the barrier type 5 is the most effective, it has not been selected in the solutions provided by the metaheuristic. This result seems counter-intuitive, but one should analyse this outcome in the light of the implementation cost. Barriers of type 5 are the most expensive, having a cost close to the maximum allowed budget. For this reason, its implementation on an edge would allow from the one hand a large increase in propagation time associated to that edge, but on the other hand, it could limit the implementation of other barriers to other fire-paths which, in the meanwhile, as a result of that allocation, might become the most critical scenarios.

To translate Figure 4.3 into a real industrial practice, the safety barriers which are now linked to edges in the figure, need to be related with nodes, or, in other words, with chemical installations. Barriers such as sprinkler systems, water deluges, or a concrete wall surrounding a tank, are applied to installations (i.e., storage tanks in our example) and not
Figure 4.3: Allocation of barriers for different values of $D$. 
Chapter 4  Mitigating domino effects by allocating protective safety barriers
to the pipework connecting the installations. Therefore, we chose the following approach. We looked at every node $i$ (with $i \in [0, \ldots, 10]$) and all its outgoing edges. We then applied the most effective safety barrier on $i$ considering the safety barriers of all the outgoing edges of $i$. This way, Figure 4.4 was developed. The symbol above each node represents the type of barrier allocated on that installation.

As can be seen in Figure 4.4, depending on the cardinality, different safety barriers are chosen for the same installations. Also, depending on the cardinality, a different total budget is needed: EUR 2.2 million in case of $D = 0$; EUR 2.15 million in case of $D = 1$; and EUR 1.9 million in case of $D = 2$. This results from the fact that one barrier linked to an installation serves several barriers linked to several edges.

The second part of the proposed approach, where we put the safety barriers on nodes, leads to suboptimal solutions. If for Figure 4.4 the $D = 2$ case is considered, we can see that certain nodes do not get applied any safety barriers. However, only EUR 1.9 million of the total safety budget is used. The translation from safety barriers on edges to safety barriers on nodes proves to be unsatisfactory. To have a more appropriate solution, the first part of the approach should be adapted to find safety barriers on nodes rather than edges. Doing so, would yield a solution approach where no transition is needed, and would consequently lead to more accurate results.

Finally, the proposed solution approach can also be used to retrieve some information concerning the resulting minimum propagation times of different fire-paths which are associated to all possible accident scenarios. As expected, the values of these times are directly re-
Figure 4.4: Allocation of barriers on each installation for different values of $D$. 
Chapter 4 Mitigating domino effects by allocating protective safety barriers

Table 4.5: Escalation Time for all fire-paths in case of different cardinalities of domino effects

<table>
<thead>
<tr>
<th>Node</th>
<th>$D = 0$ Min ET (min)</th>
<th>$D = 1$ Min ET (min)</th>
<th>$D = 2$ Min ET (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>28.05</td>
<td>54.45</td>
<td>71.95</td>
</tr>
<tr>
<td>1</td>
<td>26.40</td>
<td>43.90</td>
<td>79.20</td>
</tr>
<tr>
<td>2</td>
<td>37.70</td>
<td>55.20</td>
<td>81.60</td>
</tr>
<tr>
<td>3</td>
<td>17.50</td>
<td>43.90</td>
<td>71.95</td>
</tr>
<tr>
<td>4</td>
<td>26.25</td>
<td>52.65</td>
<td>80.70</td>
</tr>
<tr>
<td>5</td>
<td>42.90</td>
<td>69.15</td>
<td>95.55</td>
</tr>
<tr>
<td>6</td>
<td>18.15</td>
<td>44.40</td>
<td>70.80</td>
</tr>
<tr>
<td>7</td>
<td>31.50</td>
<td>49.00</td>
<td>75.40</td>
</tr>
<tr>
<td>8</td>
<td>39.00</td>
<td>57.15</td>
<td>83.40</td>
</tr>
<tr>
<td>9</td>
<td>41.25</td>
<td>74.40</td>
<td>91.90</td>
</tr>
<tr>
<td>10</td>
<td>41.25</td>
<td>74.40</td>
<td>91.90</td>
</tr>
</tbody>
</table>

lated to the characteristics of the installations and the location of the selected safety barriers.

For each root node, where the initiating event is localized, the fire-path of cardinality $D$, presenting the minimum escalation time, can be retrieved. This information is crucial for fire brigades, rescue and emergency teams since the time associated to each worst case fire-path (in term of escalation times) represents the requested maximum intervention time to stall the fire and avoid the propagation of the accident even further (i.e. domino events $> D$). An example is reported in Table 4.5, where for each installation the fire-path originating in that node with the minimum escalation time is reported.

Depending on this information the fire brigades can arrange specific
trainings and decide, in their turn, where to locate emergency facilities to reduce the intervention time and increase their effectiveness.

## 4.5 Conclusions

In this chapter, a model to support decision makers has been presented. The goal is to increase the total failure time associated to a domino event of a given cardinality which may be triggered by an accident happening to a chemical installation within a chemical plant. This is possible by investing a limited budget in safety barriers which may delay the propagation of the accident within a plant. The problem of selecting and allocating safety barriers given a limited budget, represents a complex combinatorial optimisation problem.

An iterative local search is developed to support decision makers to quickly design and solve possible accident scenarios presenting different values of domino cardinality. Depending on the cardinality of the domino event, the optimal allocation of safety barriers will change, allowing a delay of the failure time associated to the worst-case domino scenario.

The metaheuristic is applied to a realistic case study that has been designed to illustrate and prove the effectiveness of the proposed solution approach. The risk that an accident, which occurs at an installation, can trigger a propagation of that accident to other installations within the same industrial setting or to neighbouring industrial settings leading thus to domino effects, has a significant impact on the optimal allocation of safety barriers.
A pilot experiment has been conducted to determine the parameter settings of the algorithm. We tested our model and its related solution approach to an illustrative case study of storage tanks. This example is intended to show the effectiveness of the proposed decision model and the flexibility of our solution approach. The instance has been designed on the basis of validated and hence realistic data.

The decision model has been proven to be a valuable support tool to allocate protective safety barriers and mitigate the consequences of an accident engendering domino effects. Differently from a myopic optimisation, where barriers are allocated to simply prevent domino accidents, the model proposed in this chapter can be used to analyse more realistic scenarios in which domino effects need to be considered since they may have a significant impact on the allocation of protective safety barriers. The general optimisation approach, proposed in this chapter, can guide the decision maker to allocate a limited budget in order to increase the time needed to stop the escalation of an accident whose domino effects of a certain cardinality may determine the failures of other installations within the same plant or located in neighbouring plants.

The method thus can provide a valuable contribution in case of emergencies where rescue teams of fire brigades need to know the maximum intervention time that they have at their disposal to stop the escalation of the accident. The solution approach, developed in this chapter, not only is able to determine the an allocation of safety barriers to increase the intervention time in case of a domino accident of a given cardinality associated to the worst case scenario, but it can provide useful
information of the maximum intervention times to stop the escalation depending on the installation where the domino accident has originated.

For more accurate results concerning safety barriers which are placed on nodes, the solution approach should be adjusted slightly. If cases are considered where safety barriers are placed on edges, this approach can be used as is, and proves to return results of high quality.

The results obtained on the case study were quite encouraging both in term of quality of solutions and in term of flexibility. The metaheuristic can provide a solution in a limited amount of time and a large number of scenarios can be simulated by varying: (a) the cardinality of the domino accidents; (b) the available budget; (c) the features of the critical installations; (d) the number and the characteristics of the available protective barriers. For these reasons we believe that, after some more extensive testing, it can be effectively used as a powerful decision support tool not only by the decision makers to design an industrial setting considering domino effects, but also by emergency and rescue teams to evaluate the minimum time to stall the propagation of the accident depending on the features of the setting affected by the accident and the type of accident itself. In this latter case, the minimum time to intervene in case of major accident can provide valuable information on how to locate emergency or rescue facilities in order to minimise their intervention time.

In future work, the methodology proposed in this chapter should be further investigated in terms of optimality. A statistical experiment should determine the impact of variations in the budget, the topology of
the network and the properties of the safety measures. Larger instances should be constructed and the algorithm should be tested in a more rigorous way. Furthermore, an adapted algorithm (and internal data structures) should be devised and investigated to be able to cope with the allocation of safety strategies on nodes rather than edges.
Network reliability in the design of water distribution networks

This chapter is based on the manuscript “Janssens, J., De Corte, A., and Sörensen, K. (2016a). Water distribution network design optimisation with respect to reliability. Working paper, RPS-2016-007”
5.1 Introduction

With an ever growing population on our planet, the expansion and growth of cities and villages is inevitable. Together with the development and urbanisation of rural areas, comes the growth and fortification of the utility networks in those territories. To face the increase in demand, network operators can add connections to an existing network, or replace existing network sections with ones that can handle a higher throughput.

Not only do customers expect to receive a decent service, the network operator is bound by contractual limits of service variations and interruptions. But certain customers might even have contracts that enable them to file enormous claims if the contract or service limits are violated. In some cases, lives might even depend on the assurance of high-quality and reliable service, e.g., hospitals depend on clean water and a continuous provision of electricity, fire fighters depend on the continuous pressure and flow of water, etc. Failures in the network could throw a spanner in the works, and lead to catastrophic end results. Ostfeld and Shamir (1993) divide failures in water distribution networks in two closely-related categories: mechanical failure of system components and hydraulic failures in meeting customer requirements. In this work, focus will be on pipe disruptions, which falls under the first category, i.e. mechanical failures.

The cost of extending existing networks, or creating new utility networks, should not be underestimated. Network operators are pressured by share holders to keep costs at a minimum while keeping service at
an agreeable level. Network designers are faced with the problem of deciding where new lines and sections should be added or old ones should be replaced or upgraded.

The problem tackled in this work is one that can be encountered by network designers of any utility network. Where should one add lines to ensure a certain probability of service at certain high value points (critical points) in the network, at the lowest possible cost?

In the case of water network designers, an example could be the following: as a water provider, the service to a few customers in the network should be guaranteed with a probability of 99% in case of network failures. These failures can be malicious or by normal wear and tear.

In Section 5.2, the problem is explained and defined as a mathematical model. Section 5.3 gives an overview of the state-of-the-art in network planning and design. The solution approach used in this chapter, is highlighted in Section 5.4, while the results are shown in Section 5.5. Finally, in Section 5.6, the conclusions are presented.

5.2 Problem definition

In a graph $G = \{N, E\}$, where $N$ is a set of nodes and $E$ is a set of edges, a spanning tree is formed with a set of edges $\mathcal{F} \subset E$. The edges of $E$, that are not in $\mathcal{F}$ are defined as set $\mathcal{A}$, such that $E = \mathcal{F} \cup \mathcal{A}$ and $\mathcal{F} \cap \mathcal{A} = \emptyset$, the set $\mathcal{A}$ consists of the edges that can be added by the decision maker. A set of critical nodes, $K \subset N$, is also defined. A critical node is defined as a node in which the supply has to be guaranteed with a
certain probability. A critical scenario is defined as a combination of failures in edges that result in a loss of service in a node \( c \in \mathcal{K} \). \( \mathcal{C} \) is the set of all critical scenarios. The value of \( x_a \) is equal to 1 if the arc \( a \) is being added to the solution, 0 otherwise. Set \( \mathcal{C}_c \) is the set of scenarios that is critical for one critical node \( c \). Set \( \mathcal{Q}_l^F \) is the subset of edges of \( \mathcal{Q} \) that fail in critical scenario \( l \), set \( \mathcal{Q}_l^N \) is the subset of edges of \( \mathcal{Q} \) that do not fail in critical scenario \( l \), where \( \mathcal{Q} = \mathcal{F} \cup \mathcal{X} \) and \( \mathcal{X} = \{ a | a \in \mathcal{A}, x_a = 1 \} \).

With these sets defined, the problem can be represented as a mathematical model as follows:

\[
\begin{align*}
\min & \quad \sum_{a \in \mathcal{A}} x_a \cdot c_a & \quad (5.1) \\
\text{s.t.} & \quad P_c \leq M_c & \forall c \in \mathcal{K} \quad (5.2) \\
& \quad R_l = \prod_{i \in \mathcal{Q}_l^F} p_i \cdot \prod_{k \in \mathcal{Q}_l^N} (1 - p_k) & \forall l \in \mathcal{C} \quad (5.3) \\
& \quad P_c = \sum_{l \in \mathcal{C}_c} R_l & \forall c \in \mathcal{K} \quad (5.4) \\
& \quad x_a \in \{0, 1\} & \forall a \in \mathcal{A} \quad (5.5)
\end{align*}
\]

where \( c_a \) is the cost for adding arc \( a \) to the solution. The objective function in Equation (5.1) minimises the total cost of all arcs added, which is calculated as the sum of the cost of all the arcs that are added to the network. In this chapter, the cost was chosen to be equal
to the length of the arc added. A more realistic cost, e.g., which depends on the selected diameter and material, could be used, but this was not necessary to show the effectiveness and functionality of the solution approach proposed in this chapter. Equation (5.2) represents the constraint which ensures that the minimal probability of supply at the critical nodes, when a failure happens, is above a predefined percentage. $M_c$ is the maximum allowed probability of failure allowed at critical node $c$, specified by the decision maker. $P_c$ is the calculated probability of failure of critical node $c$. In Equation (5.3), we specify the probability that scenario $l$ takes place, defined as $R_l$. The probability of failure of arc $i$, if it is failing in critical scenario $l$, is represented by $p_i$. The probability of failure of arc $k$, if it is not failing in critical scenario $l$, is shown by $p_k$. Equation (5.4) defines $P_c$ as the sum of probabilities of critical scenarios happening where critical node $c$ is no longer receiving proper service. Equation (5.5) defines the domain of the decision variable, $x_a$.

5.3 Literature review

In Steiglitz et al. (1969), Grötschel et al. (1995) and Kerivin and Mahjoub (2005), a similar problem is tackled, namely, the survivable network design problem. This problem is defined on a graph $G$, where an origin node, $o$, and a destination node, $d$, are present. The survivable network design problem comes in two flavours: the node-survivable network design problem (NSNDP) and the link-survivable network design problem (LSNDP). For a network to be a link-survivable network, a network designer adds edges (links) to the network, until there is a set
of paths between $o$ and $d$ that contains at least a pre-specified number of link-disjoint paths. A link-disjoint path is defined as a path were none of the links present in the path, is present in an other path. The node-survivable network does the same with nodes-disjoint paths. In both cases, the approach neglects the real probability of failure for each component, and assumes that the probabilities for all component are equal. This, however, is an abstraction of reality, as different components could have different expected life spans and some components (e.g., moving components) could have a higher rate of failure than others. On the other hand, this abstraction does make the necessary calculations much less complicated. Both approaches also allow only one source and destination node. The method proposed in this paper is able to handle multiple source and destination nodes, and does not have an equal probability of failure for each component. Furthermore, the survivable network approach forces a pre-specified number of edge- or node-disjoint paths, what could result a in sub-optimal allocation of budget to enforce this constraint.

Reliability considerations should be present at different decision levels in water distribution companies. Nonetheless, Ostfeld and Shamir (1993) stated that very little effort is put in water distribution network optimisation that takes reliability into account as one of the objectives. One of the main causes mentioned is the difficulty in defining reliability measures. Walski (1993) appeared at the same moment as Ostfeld and Shamir (1993) and raised similar thoughts. In the following two decades, water distribution network reliability has gathered a considerable amount of attention. Still, today, it is hard to find a universally accepted definition of reliability. Concepts such as reliabil-
ity, resilience, redundancy and robustness share common grounds and are used interchangeably. Moreover, defining reliability parameters that are meaningful and measurable remains a problem. Increasing a network’s reliability can happen by taking decisions at different levels: network topology (e.g., adding new pipes), network design (e.g., changing pipe diameters) and network operation (e.g., pipe cleaning). Most authors relate topology decisions with mechanical reliability and hydraulic feasibility with design and operational decisions. This shows the importance of clearly defining what is to be studied and what the decision variables are when working on water distribution network reliability. The problem defined in this paper is clearly the first category, which concerns decisions in the network topology.

Todini (2000) aims at quantifying the concept of network reliability by using an index of resilience, which he defines as the network’s intrinsic capability of overcoming sudden failures. An increased resilience index, which is easily measurable, will lead to improved network reliability. He considers network topology as given and considers pipe diameters as decision variables. The problem is formulated as a multi-objective problem that aims at reducing the cost, while preserving a sufficient degree of resilience. He applies a heuristic approach to construct an approximate Pareto set.

Yazdani et al. (2011) study different network expansion strategies that aim at enhancing the network resilience. The authors create four different network expansions, ranging from tree-like to more looped structures. They calculate resilience metrics and how the network topology influences system robustness. No optimisation is applied. Instead, they
5.3 Literature review

use graph theoretical metrics to assess the redundancy and robustness of the network.

Ostfeld (2012) studies hydraulic feasibility in combination with the related design (pipe diameters) and operational (required pump power) decisions. They formulate the problem as a minimisation problem that is decomposed in an inner problem, finding the least cost design, and an outer problem, which is the reliability evaluation model.

Liu et al. (2014) use diameter-sensitive entropy as a surrogate measure for network reliability. They explicitly take pipe diameters into account, since diameters influence the network reliability significantly. For example, larger diameters enhance the ability to cope with abnormal demand increases (hydraulic feasibility) and they have lower probability of pipe bursts (mechanical feasibility). The problem is formulated as a multi-objective optimisation model which aims at minimizing total cost and maximising the resilience index. Pipe diameter is the decision variable (network design level).

Gupta et al. (2015) focus on level-one redundant networks, which are networks that sustain a single pipe failure without affecting consumer services. The authors apply an iterative method which uses LP to reach level-one redundant networks. The related decision variables are both on network topology level (i.e., choosing where to put parallel pipes) and network design level (e.g., selecting diameter sizes).

The approach proposed in this paper differs from the ones mentioned above. It extends the classical survivable network design problem by taking multiple source and destination nodes into account. Furthermore, probabilistic information is taken into account. The approach
proposed here is working on the network topology, and pipe diameters are not taken into account. Furthermore, in this work, the focus is on level-three redundancy.

5.4 Solution approach

Two algorithms to solve the problem described in Section 5.2 are explored. The first one is an exact solution procedure, the second one is a metaheuristic approach. Both methods make use of the same basic building blocks, such as adding and removing arcs from the solution, checking feasibility, and upper bound checking, so that spending a large amount of time in optimising one of these building blocks does not favour one approach over the other.

For both solution approaches, the decision was made to restrict the critical scenarios to those with 3 or less failing arcs. Following the terminology of Gupta et al. (2015), the aim is therefore to design level-three redundant networks. This choice can be justified by the fact that critical scenarios with 4 or more failing arcs have a very small probability of occurrence. So small even that the probability that any of these scenarios occurs, not only the critical ones, is smaller than 1.2% for the instances that are considered in this chapter. The benefit of restricting these critical scenarios to 3 failing arcs or less is mostly noticeable in the computational speed, but it also benefits the memory usage of both algorithms. Considering that in the worst case, if all scenarios, including the ones with 4 or more failing arcs, are considered, a maximum of $2^{|E|} - 1$ critical scenarios can occur, it should be clear that when
only the scenarios with 3 or less failing arcs are considered, only a fraction of this number will be possible critical scenarios. The maximum number would then be equal to \( \binom{|E|}{1} + \binom{|E|}{2} + \binom{|E|}{3} \).

### 5.4.1 Exact approach

The exact solution approach proposed here, is an approach where a recursive function iterates over all the possible combinations of additions of arcs to the network. A feasibility check on the probability constraint for each critical node (see Equation (5.2)) is done to verify if a proposed solution is feasible. The best objective function found so far is used as an upper bound throughout the algorithm to speed things up, by neglecting branches of the search tree that are feasible, but have an objective value larger than the upper bound. In Algorithm 5.1, the exact algorithm is shown in pseudo code. After initialising the parameters of the exact approach and the instance, the algorithm checks if the empty solution is feasible. If it is feasible, this is the best possible solution, as no edges have been added to the solution yet, so the objective value will be zero, the lowest possible value. If it is not feasible, the algorithm calls the recursive function shown in Algorithm 5.2, to iterate through all the combinations of possible arc additions.

The recursive function, shown in Algorithm 5.2, receives the best solution so far, \( x^* \), the current solution, \( x^f \), and the arc to add, \( \text{arcToCheck} \), as parameters. If we did not reach the end of possible arcs to add \( |\mathcal{A}| \), the arc, \( \text{arcToCheck} \), is added to the current solution. If the objective value is lower than the best objective value found so far, the probabilities for the critical nodes, given the current solution, are calculated.
Chapter 5 Water distribution network design optimisation

Algorithm 5.1 Exact approach

1: $x \leftarrow 0$, arcToCheck $\leftarrow 0$
2: $x^* \leftarrow x$, $f(x^*) \leftarrow \infty$
3: $x^t \leftarrow x$, $f(x^t) \leftarrow f(x)$
4: $P_c \leftarrow$ calculateCriticalNodeProbability($x$)
5: if $(P_c \leq M_c)$ then
6:     $x^* \leftarrow x$, $f(x^*) \leftarrow f(x)$
7: end if
8: return $x^*$
9: $x^* \leftarrow$ recursive($x^t$, $x^*$, arcToCheck)
10: return $x^*$

all the probabilities are smaller than a predefined maximum value $M_c$ for each node, the solution is stored as the new best solution found so far, and a switch, improvement, is set to true. The improvement value is only set if the objective is the lowest so far and the solution is feasible. This means that it has no use to explore that branch any further, as all arcs that get added will increase the objective value. Next, the value of arcToCheck is increased to move on to the next arc. If improvement was not set, the recursive function is called again, with the new values.

In the second part of the function, arcToCheck is decreased again, to be able to remove the arc from the current solution. If the objective value of the current solution is lower than the best objective value found so far, the feasibility is calculated again. If the current solution is feasible, it is stored as new best solution. Finally, arcToCheck is increased once more, and the recursive function is called again. At the end of the recursive function, the best found solution is returned.

The addArc function, which is used in both the exact and metaheuristic
solution methods, and the structure that holds the critical scenarios are designed to update the solution’s critical scenarios without recalculating them all from scratch. The structure for each scenario has an array of pipes that fail, the risk for that scenario, an array with the critical nodes which are disconnected from a source by this scenario, an array
Figure 5.1: Disconnection in two node groups by one of the critical scenarios

that contains arrays of node groups and two arrays to keep track of presence of a source or a critical node in a node group. A node group is a subset of nodes that are still connected to each other when the critical scenario would happen. An example is given in Figure 5.1, where in Figure 5.1(a) we can see a network where one critical scenario is indicated in red and in Figure 5.1(b) the two resulting node groups are indicated. The first node group is the group of green nodes, the second one is the set of red nodes. A clear decoupling between the two groups happens when the critical scenario would take place. The fact that at most 3 arcs can fail at the same time in a critical scenario has as a result that there can be at maximum 4 different node groups in the aforementioned scenario.

When adding an arc to the solution, we need to update the critical scenarios to keep an accurate check on the reliability in the critical nodes. Something interesting could be noticed when comparing the critical scenarios from a network before a change and the same network after
adding an arc to it. If we had a list with critical scenarios, all scenarios of the second case were based on the scenarios of the first one. This observation lead to the following algorithm for updating the scenarios after adding an arc.

When an arc is added, there are a few possible courses of action, depending on the failing edges in the critical scenarios. Two main distinctions can be made in scenarios when an arc is added. We can have scenarios where this arc will also fail and scenarios where the arc does not fail. If we take the first group, it should be clear to the reader that after the arc is added, because we consider only scenarios with at maximum 3 failing edges, from the previous critical scenarios, only scenarios with less than 3 failing arcs are plausible candidates. A copy is made and these copies are updated by adding the new arc to the set of failing arc and by multiplying the old risk with the probability of the new arc. We do not have to do any other updates, as the node groups and their respective source nodes and critical nodes stay unchanged. We can be sure that if that added arc fails, all of these critical scenarios are still critical.

The second group were scenarios where the arc is added and does not fail. In this group, different possible alternative paths of action are possible. We have to look, for each critical scenario, in which manner nodes are connected by the new arc and if an interconnection between node groups takes place. If there is no connection between node groups, the scenario is still critical, and only the risk is updated. If, however, the connection happens between node groups, the actions taken depend on whether or not a source is present in either one of them. When no water source is present in the two connected node groups, the scenario will still be critical for all critical nodes. The risk is updated, the node
groups are merged, and their respective critical nodes are updated.

If one of the node groups does have a source node, a check is done to verify if the scenario is still critical. The node groups that are not part of the connection by the new arc, if any, are checked for source and critical nodes. If a critical node is found in a node group that does not have a source node, the scenario is still critical. If it is still critical, the risk is updated, the necessary node groups are merged, and their respective source and critical nodes are updated. All non-critical scenarios are removed.

When looking into the reverse action, removing an arc from the solution, it becomes clear that not all critical scenarios can be deduced from the critical scenarios from the case where the arc is present. More specifically, when adding an arc to the network, a critical scenario with three failing edges and the added arc not failing can become not-critical. Those scenarios are removed from the list of critical scenarios. There is, however, no way to trace back this step. Those scenarios have to be regenerated (e.g., the gray critical scenarios in Table 5.1, which are a subset of the critical scenarios for Figure 5.2 without edge $g$, cannot be retrieved from the critical scenarios of the network with edge $g$. Node $A$ is the source and node $D$ is the critical node.).

However, experiments showed that this regeneration, which implies iterating over all the possibilities of 3 failing arcs, took much longer than starting from the starting solution, the spanning tree, and adding all the arcs that were not removed from the solution, including updating the critical scenarios in each step, for the instances used in this chapter.
arc was around 500 times faster than removing one. Table 5.2, shows
the comparison between the approach that removes an arc and regen-
erates the critical scenarios (denoted Remove I in the table) and the
approach where the critical scenarios are regenerated by adding arcs
(denoted Remove II). The speed-up factor (SF) denotes how many times
faster Remove II is than Remove I. For both the exact and the heur-
istic approach, we can see a significant improvement in computational
time.

5.4.2 Adaptive large neighbourhood search

To overcome the long computational times for a large part of the test
instances, a metaheuristic approach has been proposed. More specific-
ally, an adaptive large neighbourhood search (ALNS) (Ropke and Pi-
singer, 2006) heuristic is used. In the ALNS, different destroy and repair
neighbourhoods are defined, and are assigned a probability for each
neighbourhood to be selected during the search. These probabilities
are changed during the course of the algorithm to reflect the effect-
Table 5.1: Critical scenarios for $G$, with and without edge $g$

<table>
<thead>
<tr>
<th>Without $g$</th>
<th>With $g$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$de$</td>
<td>$deg$</td>
</tr>
<tr>
<td>$df$</td>
<td>$dfg$</td>
</tr>
<tr>
<td>$deh$</td>
<td>$deh$</td>
</tr>
<tr>
<td>$dfh$</td>
<td>$dfh$</td>
</tr>
<tr>
<td>$ace$</td>
<td></td>
</tr>
<tr>
<td>$acf$</td>
<td></td>
</tr>
<tr>
<td>$ade$</td>
<td></td>
</tr>
<tr>
<td>$adf$</td>
<td></td>
</tr>
<tr>
<td>$bce$</td>
<td></td>
</tr>
<tr>
<td>$bcf$</td>
<td></td>
</tr>
<tr>
<td>$bde$</td>
<td></td>
</tr>
<tr>
<td>$bdf$</td>
<td></td>
</tr>
<tr>
<td>$cdf$</td>
<td></td>
</tr>
<tr>
<td>$def$</td>
<td></td>
</tr>
</tbody>
</table>

The effectiveness of the neighbourhood in helping to improve the solution. The selection of the neighbourhood is done by a roulette wheel mechanism. If a better solution is found during the search, the probability for the used neighbourhood is increased. If the solution is not better, the probability is decreased. This cycle is repeated until a stop criterion is met. An overview of the algorithm can be found in Algorithm 5.3.

First, some instance and heuristic parameter settings are set and an initial solution is created by the randomRepair neighbourhood. The details of the randomRepair, and the other neighbourhoods used in this chapter are explained in Section 5.4.3. After an initial solution has
Table 5.2: Comparison between computational times and speed-up factor (SF) for Remove I and Remove II for both the exact and ALNS approach.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Exact Approach</th>
<th>ALNS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Remove I (s)</td>
<td>Remove II (s)</td>
</tr>
<tr>
<td>001</td>
<td>116984</td>
<td>19974</td>
</tr>
<tr>
<td>002</td>
<td>5991</td>
<td>727</td>
</tr>
<tr>
<td>003</td>
<td>8879</td>
<td>1248</td>
</tr>
<tr>
<td>005</td>
<td>518133</td>
<td>54143</td>
</tr>
</tbody>
</table>

been found, a loop is executed for as long as the stop criterion is not met. In this chapter, the stop criterion is a maximum number of iterations. Inside the loop, the score is set to the default reject score, which is a parameter passed by the user, and a destroy and repair neighbourhood are selected through the use of the roulette wheel selection method.

The roulette wheel selection mechanism, also known as fitness proportion selection, is a very simple selection mechanism where one of the options is selected by generating a random number between zero and one. The probabilities assigned to the options are scaled by the total sum of probabilities so their sum becomes equal to one. Next, the probabilities are subtracted one by one, until the random number becomes negative. Once the random number becomes negative, the option to which the last subtracted probability belonged, is the selected option.

The destroy and repair neighbourhoods that were selected are explored, and the $\Delta_e$ value is then calculated based on the previous and new objective value. If $\Delta_e$ is smaller than zero, it means that the solution found by the neighbourhoods is worse than the previously found solu-
Algorithm 5.3 Adaptive large neighbourhood search

1: \( x \leftarrow \text{randomRepair()} \)
2: \( x^* \leftarrow x, f(x^*) \leftarrow f(x), \rho^- \leftarrow (1,\ldots,1), \rho^+ \leftarrow (1,\ldots,1), T \leftarrow 25, T_\alpha \leftarrow 0.9 \)
3: iteration \leftarrow 0
4: \textbf{while} (iteration \neq \text{MaxIterations}) \textbf{do}
5: \( s \leftarrow \text{rejectScore} \)
6: \( d \leftarrow \text{rouletteWheelSelection}(\rho^-, \Omega^-) \)
7: \( r \leftarrow \text{rouletteWheelSelection}(\rho^+, \Omega^+) \)
8: \( x^t \leftarrow r(d(x)) \)
9: \( \Delta_e = f(x) - f(x^t) \)
10: \textbf{if} \( \Delta_e < 0 \) \textbf{then}
11: \hspace{1em} \textbf{if} \( (e^{(\Delta_e/T)} > \text{random}(0,1)) \) \textbf{then}
12: \hspace{2em} \( x \leftarrow x^t \)
13: \hspace{2em} \( T \leftarrow T \ast T_\alpha \)
14: \hspace{2em} \( s \leftarrow \text{acceptScore} \)
15: \hspace{1em} \textbf{end if}
16: \textbf{end if}
17: \textbf{if} \( f(x^*) > f(x^t) \) \textbf{then}
18: \hspace{1em} \( x^* \leftarrow x^t \)
19: \hspace{1em} \( s \leftarrow \text{improvementScore} \)
20: \textbf{end if}
21: \( \rho^-_d \leftarrow \delta \rho^-_d + (1 - \delta)s \)
22: \( \rho^+_d \leftarrow \delta \rho^+_d + (1 - \delta)s \)
23: iteration \leftarrow iteration + 1
24: \textbf{end while}
25: return \( x^* \)

If this is the case, the solution is accepted with a certain probability. This probability is expressed as:

\[ P = e^{(\Delta_e/T)} \] (5.6)

where \( T \) is the temperature. When the solution is accepted, the tem-
perature is updated with a predefined value $T_a$. The score $s$ is updated to the accept score. This acceptance method is borrowed from the simulated annealing heuristic, which was introduced by Kirkpatrick et al. (1983).

If the objective value of the current solution is better than the best found objective value so far, the solution is accepted as the new best solution. The score $s$ gets updated to the improvement score. After that, the probabilities of the destroy neighbourhood and repair neighbourhood that were selected ($\rho_d^-$ and $\rho_r^+$ respectively), get updated depending on the score $s$ that was set, and a decay value $\delta$.

\begin{align}
\rho_d^- &\leftarrow \delta \rho_d^- + (1 - \delta)s \\
\rho_r^+ &\leftarrow \delta \rho_r^+ + (1 - \delta)s
\end{align}

The value of the score determines if the probability of selection of the used destroy and repair neighbourhoods becomes larger or smaller. At the end of the loop the iteration counter is updated. When the stop criterion is met, the best solution is returned, and the ALNS heuristic is finished.

5.4.3 Neighbourhoods

Two main classes of neighbourhoods have been used. A random type, where random arcs are removed from or added to the solution for the random destroy and random repair neighbourhood respectively, and
Chapter 5 Water distribution network design optimisation

Algorithm 5.4 RandomRepair function

1: \( m \leftarrow |\text{unused arcs}| \)
2: \( c \leftarrow m, \text{stop} \leftarrow \text{false}, i \leftarrow 0 \)
3: \textbf{for} \((i < m \text{ and } !\text{stop}) \) \textbf{do}
4: \( a \leftarrow \text{random()} \mod c \)
5: \( x^t \leftarrow \text{addArc}(x^t, a) \)
6: \( P_c \leftarrow \text{calculateCriticalNodeProbability}(x^t) \)
7: \textbf{if} \( (P_c \leq M_c) \) \textbf{then}
8: \( \text{stop} \leftarrow \text{true} \)
9: \textbf{end if}
10: \text{decrease} \ c, \text{increase} \ i
11: \textbf{end for}
12: \text{return} \ x^t

the well known greedy randomized adaptive search procedure (GRASP) type, where arcs that are selected through a roulette wheel selection based on a ranking are added or removed. Both neighbourhoods are used in different forms depending on the number of edges to be removed, or the ranking used to feed the roulette wheel selection mechanism.

The randomRepair is the only exception, as there was only one variant used. Its pseudo code is shown in Algorithm 5.4. While the solution found is not feasible, the random repair will select an unused arc at random and add it to the solution. Next, its feasibility is checked. Once the solution becomes feasible, or there is no more arc to add, the solution is returned.

The randomDestroy neighbourhood, is very similar. For that neighbourhood, a percentage of assigned arcs to be removed from the solution is defined. This number of arcs is then removed from the solu-
5.4 Solution approach

Algorithm 5.5 GRASPRepair function

1: \( m \leftarrow |\text{unused arcs}| \)
2: \( c \leftarrow m, \text{stop} \leftarrow \text{false}, i \leftarrow 0 \)
3: \textbf{for} (\( i < m \) and \( \text{!stop} \)) \textbf{do}
4: \( p \leftarrow \text{top(SortedPriorityList)} \)
5: \( a \leftarrow \text{rouletteWheelSelection}(p) \)
6: \( x' \leftarrow \text{addArc}(x', a) \)
7: \( P_c \leftarrow \text{calculateCriticalNodeProbability}(x') \)
8: \textbf{if} (\( P_c \leq M_c \)) \textbf{then}
9: \( \text{stop} \leftarrow \text{true} \)
10: \textbf{end if}
11: decrease \( c \), increase \( i \)
12: \textbf{end for}
13: return \( x' \)

tion, by randomly selecting arcs to remove. Different percentages were defined.

The second type of destroy and repair neighbourhoods are all based on the GRASP approach. The GRASPRepair method is shown in Algorithm 5.5. The main difference with the random approach, is that the arcs are selected from a sorted priority list, by using a roulette wheel selection mechanism, which takes in to account the probabilities of the different alternatives.

For the GRASPRepair method, 5 different variants are defined. The variants are using different priority lists, namely, \textit{cost}, \textit{reliability}, \textit{1/cost}, \textit{1/reliability} and \textit{reliability/cost}. An overview can be found in Table 5.3.

The GRASPDestroy method uses the same priority rankings. However, they also have a parameter for the percentage of arcs to remove,
resulting in 20 different GRASPDestroy variants. An overview of the different destroy neighbourhoods is given in Table 5.4.

After the analysis of the results obtained by the adaptive large neighbourhood search (ALNS), it is clear that suboptimal results are obtained. Often, the optimal solution is contained in the solution found by the ALNS, but a large number of excess arcs are present in the final solution. This can be explained by the approach used in the repair operators. The repair operators continue to add arcs to the solution until the reliability threshold is met. The sorting of the arcs to be added might not be the sorting which yields the optimal results. To improve the performance of the ALNS algorithm, a local search based cleanUpSolution method is proposed, that will iterate over the arcs in the solution, and will check if the solution is still feasible if that arc is removed. If the solution is still feasible, the arc is removed from the solution. Figure 5.3 gives an overview of the results obtained by the ALNS algorithm with application of the cleanUpSolution method for different numbers of iterations on one of the instances used in this chapter. When using 250 iterations, 90% of experiments ran with the ALNS algorithm found the optimal solution as opposed to none when the cleanUpSolution method was not applied.

<table>
<thead>
<tr>
<th>Type</th>
<th>Variant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>-</td>
</tr>
<tr>
<td>GRASP</td>
<td>cost, reliability, $\frac{1}{\text{cost}}$, $\frac{1}{\text{reliability}}$, reliability $\text{cost}$</td>
</tr>
</tbody>
</table>
Table 5.4: Overview of destroy neighbourhoods

<table>
<thead>
<tr>
<th>Type</th>
<th>Variant</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>-</td>
<td>10, 20, 30, 40</td>
</tr>
<tr>
<td>GRASP</td>
<td>cost, reliability, $\frac{1}{\text{cost}}$, $\frac{1}{\text{reliability}}$, reliability, $\frac{\text{reliability}}{\text{cost}}$</td>
<td>10, 20, 30, 40</td>
</tr>
</tbody>
</table>

Figure 5.3: Results for ALNS with cleanUpSolution method

5.5 Computational experiments

5.5.1 Instance generation

The set of test networks that is used in this work is based on networks created by HydroGen (De Corte and Sörensen, 2014). HydroGen is a tool to generate realistic water distribution networks of arbitrary size and characteristics. The generated networks are available in EPANET (Rossman (2000)) input format, since this is the most frequently used
hydraulic solver, and GraphML, an XML-based graph exchange file format, which enables easy visualisation. A database of these networks is available online (http://antor.uantwerpen.be/hydrogen).

The generation method follows different phases. In a first step, random cluster centres are generated in a two dimensional plane. A minimal and maximal distance between these centres can be specified. The water demand nodes are constructed in a circular layout around these cluster centres. Next, a minimum spanning tree, connecting all of the demand nodes in the plane, is drawn, using Prim’s algorithm (Prim, 1957). These connections represent the water distribution pipes. Every pipe has a begin-node, end-node, length, diameter and roughness coefficient. Begin-node, end-node and length are defined by the generation itself: nodes are assigned while drawing the spanning tree and the edge weights or pipe lengths are the Euclidean distances. These edges or pipes define the set $\mathcal{F}$.

For every reservoir or tank that has to be added to the water distribution network, a random demand node at the outside of the cluster is selected, and the water supply is connected to this demand node with a new pipe or pump. Although a tree structure could efficiently provide every demand node with sufficient drinking water, no real-life water distribution network is designed as a tree. Therefore, potential loops are added to increase water delivery reliability. The pipes that create these loops define the set $\mathcal{A}$. Moreover, huge pressure changes are avoided by an interwoven net. Duplication of pipes and the crossing of pipes are avoided by a preliminary check. Moreover, in reality, the number of neighbouring pipes (or the so-called degree of that node) is
5.5 Computational experiments

Figure 5.4: Example of a HydroGen test instance (left) and a zoom-in on part of this instance (right)

limited to four, which is also taken into account in the generation procedure. As stated before, for some nodes in the network, it is of crucial importance that a continuous provision of drinking water is guaranteed. These critical nodes only allow a certain probability of failure. In a final step of the network generation procedure, it is determined which nodes of $\mathcal{N}$ are critical ($\in \mathcal{K}$) and which ones are not.

An abundant number of realistic network settings can be built by adjusting HydroGen’s parameters. By fine-tuning these parameters, one can obtain networks that have a high resemblance to real networks.
Table 5.5: Legend symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>□</td>
<td>Reservoir</td>
</tr>
<tr>
<td>•</td>
<td>Critical node (\in \mathcal{K})</td>
</tr>
<tr>
<td>•</td>
<td>Demand node (\in \mathcal{N})</td>
</tr>
<tr>
<td>----</td>
<td>Potential edge (\in \mathcal{A})</td>
</tr>
<tr>
<td></td>
<td>Edge (\in \mathcal{F})</td>
</tr>
</tbody>
</table>

5.5.2 Statistical analysis

A full factorial experiment has been conducted with the parameter settings shown in Table 5.6. We used R (R Core Team, 2015) and the *lme4* package (Bates et al., 2015) to perform a linear mixed effects analysis of the relationship between the parameter values and the objective value and CPU time, taking the instance as a random effect by following the step by step analysis as shown in Winter (2013).

Two instances (007.inp and 010.inp) were quite reliable on their own and already had a 95% reliability. All instances were run for 95% reliability, and therefore those two instances did not yield any useful results. They are removed from the analysis.

In a first visual inspection of the residual plot, we could see that heteroscedasticity might be a problem for both objective value and computational time. To handle that, we did a log transform of the objective value and of time and this seemed to solve the problem for both.
5.5 Computational experiments

Table 5.6: Overview of the parameter settings for the full factorial experiment

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Tested values</th>
<th>Selected value</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRASP list size (g)</td>
<td>1, 2, 3, 4</td>
<td>4</td>
</tr>
<tr>
<td>Maximum number of iterations (m)</td>
<td>10, 50, 100</td>
<td>100</td>
</tr>
<tr>
<td>Accept score (a)</td>
<td>0.8, 1, 1.2</td>
<td>1.2</td>
</tr>
<tr>
<td>Reject score (r)</td>
<td>0.4, 0.5, 0.6</td>
<td>0.6</td>
</tr>
<tr>
<td>Improvement score (i)</td>
<td>1.5, 2, 2.5</td>
<td>1.5</td>
</tr>
<tr>
<td>Decay value (d)</td>
<td>0.3, 0.6, 0.9</td>
<td>0.9</td>
</tr>
</tbody>
</table>

The ANOVA shows that both GRASP list size and number of iterations have a significant impact on the objective value. They both lower the objective value if their values get increased. For GRASP list size, this is significant with $\chi^2(3) = 8.4214, p = 0.03806$, and for number of iterations, this is significant with $\chi^2(2) = 539.37, p < 2.2e^{-16}$.

For time, the GRASP list size, the number of iterations and the decay value are significant. When the maximum number of iterations increases, the time increases with it (which makes perfect sense). All the other parameters did not seem to make a significant difference in objective value or running time.

In Figure 5.5 and Figure 5.6 the mean plots for each parameter for each instance is shown with respect to the log transformed objective value and CPU time. From these figures and the mean plot for each parameter grouped over the instances, the parameter settings shown in Table 5.6 are selected.

For the significant parameters for both objective value and CPU time,
Table 5.7: Results of the exact approach compared with those of the metaheuristic

<table>
<thead>
<tr>
<th>Instance</th>
<th>Exact Approach</th>
<th>Metaheuristic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Best Solution</td>
<td>Time (s)</td>
</tr>
<tr>
<td>001.inp</td>
<td>68.94</td>
<td>13460</td>
</tr>
<tr>
<td>002.inp</td>
<td>51.69</td>
<td>537</td>
</tr>
<tr>
<td>003.inp</td>
<td>32.29</td>
<td>1068</td>
</tr>
<tr>
<td>004.inp</td>
<td>120.85</td>
<td>58591</td>
</tr>
<tr>
<td>005.inp</td>
<td>127.20</td>
<td>37709</td>
</tr>
<tr>
<td>006.inp</td>
<td>180.20</td>
<td>14551</td>
</tr>
<tr>
<td>008.inp</td>
<td>202.70</td>
<td>18790</td>
</tr>
</tbody>
</table>

the best in terms of objective value are chosen. For decay value, the best value in terms of CPU time is chosen. For the non significant parameters, we selected the value which had the lowest mean objective value. An overview of the alternatives and the selected values are shown Table 5.6.

Table 5.7 gives an overview of the results found by the naive exact approach and the ALNS algorithm, run with the selected parameter settings. The ALNS found the optimal solution in 86% of the runs with the proposed parameter settings, and for most instances on average the time is lower than the time needed by the exact approach. In the full factorial experiment, 80% of the runs with the ALNS found the optimal solution.
5.5 Computational experiments

Figure 5.5: Plot of average log transformed objective values and cpu times for given parameter setting, grouped by instance
Figure 5.6: Plot of average log transformed objective values and cpu
times for given parameter setting, grouped by instance (part 2)
5.6 Conclusion and discussion

We have proposed a mathematical model of the water distribution network design problem where the goal is to optimise the reliability in the network given a limited budget. An adaptive large neighbourhood search solution method with different destroy and repair neighbourhoods is proposed and compared to an exact approach to solve the model.

From the analysis, we can see the algorithm is very effective. It finds the optimal solution 86% of the time when ran with 100 iterations in a very short amount of time (on average 2705 seconds). We have found that, when the number of iterations is increased, the algorithm will find the optimal solution more often, but the computational time will also increase. Also the grasp list size had an influence on the objective value and computational time.

It was clear from our analysis that the most powerful addition to the algorithm was the cleanUpSolution method. This method made the difference between finding the optimal solution or not. With the data retrieved from the algorithm, we were unable to verify if specific destroy or repair neighbourhoods had a significant impact on the result. Future research should be conducted to be able to identify key neighbourhoods. One approach would be to include or exclude the neighbourhoods one at a time to verify if the optimal result can be found.
Part II

ROUTING FOR COURIERS
Introduction
Courier companies deal with very large sets of customers and vehicles on a daily basis. Coping with this amount of locations can turn out to be very difficult. To take care of this, courier companies sometimes divide the distribution area into work areas, with each its own route plan, based on geographical location and zone numbers. As the workload in each area can differ from day to day, this does not always result in an equal distribution of workload amongst the different drivers. To handle these daily fluctuations, a similar approach is taken, but it considers smaller working areas, which can be redistributed between drivers. Three main points of focus for courier companies came to mind when doing this research. An equal work division between drivers, driver familiarity with the area and returning clients, to get a faster and more reliable delivery and the shortest possible total driving distance to keep the operating costs as low as possible.

This approach would start by dividing the work area in smaller zones of equal amounts of work, based on historical data. These zones would get distributed equally amongst the different drivers, in a smart way (taking into account geographical locations), which would result in a tactical plan. This tactical plan is the starting point for the daily operational plan.

In Chapter 7, a way to get an economically interesting operational planning, while maintaining a balanced workload between drivers and guaranteeing a high driver familiarity with the area he is working in, is investigated. The problem and an efficient metaheuristic approach to solve it are proposed.

Finally, in Chapter 8 the effect of the algorithmic parameters and the
interaction effects of these parameters with the instance characteristics on the quality of the Pareto optimal solutions found by the aforementioned algorithm are analysed. The PROMETHEE method is used to retrieve a ranking of the possible alternatives. Which in turn is analysed with ANOVA.
Multi-objective microzone-based vehicle routing for courier companies: From tactical to operational planning

7.1 Introduction

Following the example of FedEx, many courier companies now organise their distribution system as a hub-and-spoke model (see FedEx (2006), Rodrigue et al. (2006), Waldheim and Berger (2008) and Lyster (2012)). In this model, vehicles collect parcels and bring them to the nearest depot where they are consolidated and transported, through a chain of hubs, to the depot closest to their final destination. From this final depot, the parcels are brought to their destination by vehicles performing so-called milk-runs.

This chapter focuses on the last step: the delivery process from the final depot of a courier company to the end customers. On a daily basis, courier companies face a vehicle routing problem (VRP) that is not only among the largest in the logistics industry, but also highly dynamic. It is not uncommon for a courier company to execute several thousand stops from a single depot, using several hundred vehicles. Additionally, the list of pick-up locations changes throughout the day as customers call to order the pick-up of a parcel, and the list of drop-off locations is only available a very short time before the vehicles leave the depot. In order to deliver these parcels in an efficient manner, route planning software could in theory provide efficient solutions, but experience shows that such software is rarely used in practice.

The fact that the distribution planning process in many courier companies is not automated can be ascribed to a number of reasons. Firstly, correct data on exact drop locations is often lacking, i.e., customer addresses can be incomplete due to sloppy writing, carelessness, or lack
of information on behalf of the sender. Because deciphering these addresses is best done by people with an understanding of the geographical distribution region, this task is often left to the drivers or to other people in the final depot. Couriers, therefore, lack the possibility to generate the routes algorithmically without considerable investment in a centralized address-scanning-and-correcting service. Secondly, conventional route planning software requires all data to be available and correct before a final routing plan can be built. This implies that processes that depend on the final route plan, such as sorting and loading parcels into the correct vehicles, have to wait until all order entries are known. As this would extend the delivery time, one of the main competitive factors for a courier, it is considered an unattractive option.

The surplus in route efficiency achieved by route planning software would not compensate for the increased delivery time and the loss of flexibility, couriers rely on more stable organizational structures for the distribution of their parcels. A popular technique to deal with this flexibility requirement is to partition the area serviced from the depot into regions, usually one per vehicle. All customers that fall within a region are then assigned to the vehicle this region belongs to, which implies that the sorting of parcels per vehicle can start as soon as the parcel arrives at the depot. The final routing, i.e., determining the order in which to visit the customers in his\(^1\) region, is then left to the driver. This way of working has the additional advantage that a driver has the possibility to get acquainted with his region.

Figure 7.1 shows the parcel distribution process from the arrival of the

\(^1\) To avoid awkward formulations like his/her, we will refer to drivers in the masculine form and decision-makers/planners in the feminine form.
7.1 Introduction

Order intake
Routing
Sorting
Loading
Time
(a) Time when processed based on destination address

Order intake
Routing
Loading
Time
(b) Time when processed based on zones

Figure 7.1: The parcel distribution process from the arrival of the parcels, until the departure of the vehicles.

parcels, until the departure of the vehicles. Figure 7.1(a), shows the time consumed if all parcels are processed based on their destination address. Figure 7.1(b), shows the time consumed if parcels are processed based on zones. If zones are used, the sorting of the parcels can start much earlier, which results in a reduction of time from the arrival of the parcels until the departure of the vehicle.

In case of workload imbalances or vehicle capacity violations, drivers are sometimes allowed to swap individual packages among themselves. For this reason, the physical layout of the distribution depot sometimes resembles that of the geographical delivery region, in the sense that vehicles that serve adjacent routes are stationed close to each other.

To diminish the main disadvantage of zone-based organization — the limited possibility to automate the planning process — several authors have proposed an alternative method, in which the geographical region is divided into smaller zones (“microzones”). In practice, each microzone corresponds to a bin or container into which the parcels can be dropped at the moment they arrive at the final depot.

To ensure stability in the planning, the preferred assignment of micro-
zones to vehicles is fixed in a tactical plan. When an estimate of the workload in each microzone becomes available, the microzones are re-assigned to the different available vehicles, with a preference for the vehicle they are assigned to, in the tactical plan. The exchange between vehicles no longer happens on the level of individual parcels, but on the microzone level. In this chapter, we present an approach to assign microzones to vehicles and determine the order in which the microzones should be traversed. We model this problem as a tri-objective vehicle routing problem and develop an efficient multi-objective metaheuristic to solve it.

This chapter is organised as follows: Section 7.2 gives a short literature overview, followed by the problem delineation in Section 7.3. Section 7.4 elaborates on the approach we used, the neighbourhoods that are explored and the algorithm we developed. Finally Section 7.5 presents the results of our experiments. Our findings and conclusions as well as our ideas on future research are posed in Section 7.6.

### 7.2 Literature review

Although the microzone concept has been described by different names ("zones", "cells", "core areas", "flex zones"), the advantages of using microzones as building blocks for routes in large-scale vehicle routing problems have been recognised by several authors (Zhong et al., 2007; Mourgaya, 2004; Mourgaya and Vanderbeck, 2006, 2007). A complete application of this concept requires solving several challenging problems (Expósito-Izquierdo et al., 2016; Expósito-Izquierdo et al., 2013):
(1) creating robust microzones, (2) assigning microzones to routes in a tactical plan, (3) estimating the workload in each microzone and each route (Daganzo, 1984), (4) reassigning microzones to vehicles in balanced routes on an operational level, and (5) determining the final route planning within the microzones.

In this overview, we focus on the fourth challenge, the operational assignment of microzones to vehicles and the sequencing of the microzones for each vehicle, taking into account the tactical plan. Previous research has focused on the use of a tactical assignment of zones to routes with the purpose of balancing these routes and allowing drivers to familiarize themselves with certain parts of their tours. Less variability allows drivers to gain experience in their region and improve the relationship with regular customers (see Kunkel and Schwind (2012)).

In this research, we tackle this problem, by using a multi-objective approach. More specifically, we address the problem of transforming unbalanced tactical route plans into balanced and feasible routes, where the imbalance in the tactical plan is due to the stochastic nature of the problem.

Both Zhong et al. (2007) and Mourgaya (2004) try to improve package delivery under uncertainty by building a robust dispatching system, which determines flexible routes, with drivers who are more familiar with their zones, in order to provide better service and lower costs. Ouyang (2007) has done research concerning the design of zones for large-scale distribution systems.

The method proposed in Zhong et al. (2007) tries to find a balance
between driver familiarity with a zone, this can be accomplished by assigning the same driver to the same service area each day, and the flexibility to optimise dispatch plans on a daily basis, which is accomplished by adjusting the number of vehicle routes, and maximizing driver utilization. They propose using core areas, which are serviced by the same driver every day, in combination with a flex zone and unassigned zones, where customers are serviced by a different driver on a daily basis depending on the utilization of the vehicles in the neighbourhood of that cell. The authors further develop a constructive heuristic in which cells are added to the partial routes, minimizing time duration of the routes, and taking into account driver familiarity with a cell.

Similar to portfolio management, where combining uncorrelated assets leads to a lower portfolio risk (see Bodie et al. (2005)), to decrease the imbalance of the workload for each vehicle in the tactical plan, a combination of zones will be taken (as the workload for the zones is not correlated).

Mourgaya and Vanderbeck (2006) approach a multi-periodic problem by using a tactical plan optimizing both the workload balance as the regionalisation of the routes. They cluster clients based on the distance between them. The clustering is based on well known classification models.

Several papers have tackled the issue of workload balancing between the vehicles of a vehicle routing problem (Nikolakopoulou et al., 2004; Liu et al., 2006; Santiago et al., 2013).

Like many related problems found in the literature, the problem de-
veloped in this chapter is a multi-objective optimisation problem. Unlike single objective optimisation, multi-objective optimisation leads to a set of solutions known as the Pareto-optimal set. This set consists of non-dominated solutions, i.e., solutions for which no other solution exists that has at least the same performance on all objectives and a better performance on at least one. For a more formal definition of dominance we refer the reader to Ehrgott and Gandibleux (2000).

For multi-objective combinatorial optimisation problems (MOCO) like the problem solved in this paper, finding the efficient frontier by means of exact methods, is computationally very expensive and sometimes even impossible (Caballero et al., 2007; Zhiping and Yuxing, 2010). For that reason, metaheuristic algorithms become an attractive choice (Ehrgott and Gandibleux, 2000). For courier companies, finding a small set of good solutions in a short computational time is more important than finding the optimal Pareto set.

A common approach to approximate the Pareto set in multi-objective optimisation problems, is to optimise the weighted sums of objectives for varying weights (Marler and Arora, 2004; Konak et al., 2006). The main drawback with this varying weights approach is that not all Pareto-optimal points can be found when the Pareto front is not convex, i.e., it only finds the solutions on the convex hull of the optimal Pareto set (see Geiger (2001) and Jozefowicz et al. (2008)). Nevertheless, with regard to our goal, it remains an attractive technique as it is straightforward, guarantees that the solutions converge towards the Pareto-front, and is computationally efficient (Konak et al., 2006). Another important advantage is that it allows employing single-objective
heuristics or metaheuristics. Using different weight combinations will guide the search in different directions, leading to a more diversified result set, which is either the true shape of the Pareto-optimal set, or a convex approximation of it (see Caramia and Dell’Olmo (2008) and Jozefowiez et al. (2008)). The use of a different multi-objective optimisation method, or a true multi-objective metaheuristic could be helpful to give a better estimate of the real Pareto set (Geiger, 2001).

The methods defined in Zhong et al. (2007) and Mourgaya and Vanderbeck (2006) use a constructive heuristic to create the tactical and operational plan. The focus, however, is more on the creation of the tactical plan. In this chapter, we present a variable neighbourhood tabu search heuristic which is focused on rebalancing an existing tactical plan. Variable neighbourhood search (VNS) is a well-known technique in which different neighbourhoods are used systematically (see e.g., Hansen and Mladenović (2001), Imran et al. (2009) and Fleszar et al. (2009)). Tabu search is an approach used to diversify the search by excluding previously found solutions or previously executed moves (see e.g., Brandão (2004)).

To compare different Pareto-optimal sets, we will make use of the hypervolume (see Okabe et al. (2003)). The hypervolume for two or three objectives can be seen respectively as the area or volume between a reference point and the Pareto-front. Zitzler defined it as the size of the dominated space (see Zitzler and Thiele (1998a, 1999)). This means that the larger the hypervolume, the closer the found solution set is to the theoretical optimal solution set.
7.3 Problem definition

The vehicle routing problem developed in this chapter is defined on a graph $G = (N, E)$. The nodes of this graph represent the microzones, i.e., geographically distinct regions that are part of the distribution area. The set of microzones is labeled $Z$. An extra node represents the depot $o$, from which all vehicles depart and where they all arrive. Together, the microzones and the depot form the set $N = Z \cup \{\text{depot}\}$. A set of vehicles $V$ is also specified, where $|V| = R$ is the number of available vehicles. A tactical plan is given, in which each microzone is assigned to exactly one vehicle. This is represented by the parameter $y_{ik}$ which is 1 if microzone $i \in Z$ is assigned to vehicle $k \in V$ in the tactical plan and 0 otherwise. The working time $t_i$ in a microzone $i$ is a number that represents an estimate of the time to deliver all parcels in that microzone and includes the (estimated) intra-zone driving times.

The edges in the edge set $E$ of graph $G$ represent the travel between zones. The time a vehicle takes to drive between microzones $i$ and $j$ is represented as $t_{ij}$. We use the travel time between the center points of microzones as an estimate for this driving distance. It is a rough estimate, but it will stay constant as long as the same microzones are considered. The real driving distance can vary a lot based on the distribution of the delivery and pick-up locations inside a microzone and based on the position where the microzone is entered. We make the assumption that individual addresses of customers in a microzone are not known and are volatile over time. Therefore, the actual distance between the closest points of the microzones is not known.
and is, hence, not a suitable measure for the driving distance between microzones.

The aim of the problem is to find an operational route plan, i.e., an assignment of microzones to vehicles, as well as an order in which each vehicle should visit the microzones assigned to it. An operational route plan is determined by decision variables $y_{ik}$ which take a value 1 if microzone $i$ is assigned to vehicle $k$ and 0 otherwise, and $x_{ij}$, which is 1 if a vehicle travels directly between microzones $i$ and $j$ and 0 otherwise.

This problem has three objectives:

- **Objective 1 - Minimise total transportation cost**: measured by total travel time.

  \[
  \min f_1 = \sum_{i \in N} \sum_{j \in N} t_{ij}x_{ij} \tag{7.1}
  \]

- **Objective 2 - Minimise deviation from tactical plan**: measured by counting the microzones that are assigned to a different vehicle.

  \[
  \min f_2 = \frac{1}{2} \sum_{i \in \mathcal{I}} \sum_{k \in \mathcal{V}} |y_{ik} - \bar{y}_{ik}| \tag{7.2}
  \]

- **Objective 3 - Minimise workload imbalance**: measured by the square root of the sum of the squared deviations from the average route time.
Problem definition

\[
\min f_3 = \sqrt{\frac{1}{R} \sum_{k \in V} (l_k - \frac{1}{R} \sum_{k \in V} l_k)^2}
\]  \hspace{1cm} (7.3)

where \(l_k\) is the workload of vehicle \(k\) and is defined as the sum of the (estimated) inter-zone driving times plus the working time for the assigned microzones.

\[
l_k = \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} y_{ik} x_{ij} t_{ij} + \sum_{i \in \mathcal{Z}} y_{ik} t_i , \forall k \in \mathcal{V}
\]  \hspace{1cm} (7.4)

For each vehicle \(k\), its total working time in the operational route plan is limited to a pre-specified value.

\[
\forall k \in \mathcal{V} : l_k \leq W
\]  \hspace{1cm} (7.5)

For reasons of simplicity, we assume in this chapter that both the preferred working time and the maximum allowed working time are the same for all vehicles (drivers), something which is definitely not the case in real life. However, the model can be easily adapted to such situations. The constraint in Equation (7.5) can be changed to have a different maximal working time per vehicle by replacing the global constant \(W\) with a maximal allowed working time per vehicle \(W_k\). Objective 3, the minimisation of the workload imbalance, can be modelled in different ways when working times are variable. One way is to minimise the
workload imbalance separately for different values of expected workload (e.g., 8h and 4h). Another way would be to normalize the data, i.e., divide $l_k$ by $W_k$ and minimise the sum or squared sum of deviations of $l_k / W_k$ from 1.

Another assumption of our model is that the vehicles are not capacity-constrained. This is a realistic assumption, given that many courier companies only distribute relatively small parcels, letters, etc.

Except for the two additional objectives, the problem developed in this chapter is a distance-constrained vehicle routing problem (without capacity constraint). In the next section, we develop an efficient algorithm for this problem.

### 7.4 Methodology

Here, we develop a simple yet effective metaheuristic to solve the multi-objective vehicle routing problem described in Section 7.3. The algorithm, presented in this chapter, is able to produce a set of non-dominated solutions, and it is left to the decision maker to pick the final solution she decides to implement.

Our algorithm constructs a new routing plan for the multi-objective VRP by iteratively applying a multi-neighbourhood tabu search heuristic to a weighted exponential sum of the objective functions discussed in Section 7.3. This heuristic uses a set of problem-specific operators to expeditiously search for high-quality solutions.
7.4 Methodology

To efficiently evaluate intermediate solutions, generated using the tabu search heuristic, a weighted exponential sum of normalized values of the tree objective functions is used:

$$f(x) = \left[ w_1 f_1^n(x)^P + w_2 f_2^n(x)^P + w_3 f_3^n(x)^P \right]^{\frac{1}{P}}$$  \hspace{1cm} (7.6)

where $w_1 + w_2 + w_3 = 1$.

By varying the weights of the weighted exponential sum, the tabu search heuristic is forced to explore different areas of the search space, resulting in diverse solutions. Moreover, the value of $P$ allows to adjust the curvature of the objective function, which results in a better fit with the Pareto front, and which allows to capture efficient points on the non-convex part of the Pareto front (Messac et al., 2000b,a; Marler and Arora, 2004). This type of objective function is easy to implement, and allows to prioritize certain types of solutions by changing the values of the weights.

If the value of $P$ is not large enough, however, not all Pareto-optimal points can be found when the Pareto front is not convex. Nevertheless, with regard to our goal, it remains an attractive technique as it is straightforward and computationally efficient.

7.4.1 Neighbourhoods

Like a majority of efficient algorithms to solve vehicle routing problems, our variable neighbourhood tabu search uses several different
neighbourhoods to escape from local optima and overcome the myopic behaviour of a single neighbourhood.

To speed up the search process, the moves are restricted to operate on so-called “natural neighbours”. Natural neighbours are used to limit the search space by excluding moves that could never yield any positive results. It should be clear that microzones that are not near to each other (e.g., on the opposite side of the area that is serviced) can be changed, but that such a change will give an undesirable result, as it is not efficient at all. By limiting the search space to only natural neighbours, we might exclude the optimal solution, but will reduce the time for exploring the search space and finding a very good solution. Natural neighbours are defined as nodes that are connected in a Delaunay triangulation of the original fully connected graph on which the problem is defined. The Delaunay triangulation was first defined by Delaunay (1934) as a triangulation of the points in the system, in such a way that the circumcircle of any triangle does not contain any other points of the system.

We consider only euclidean instances in this chapter. If other instances are to be considered, a projection can be made into euclidean space and the Delaunay triangulation can be performed on that projection. If no projection can or should be made, another method to find nearest neighbours is to be used. One possibility is to use the distance matrix to select $n$ nearest neighbours to evaluate.

An alternative procedure to restrict the search space of the neighbourhood operators, would be to limit the moves to pairs of microzones that
are not more than a certain maximum distance apart. The main drawback of this approach is that this maximum distance needs to be determined, an arbitrary decision which may be difficult when the nodes are not evenly spaced. In such situations, microzones corresponding to remote areas may feature in much fewer moves than microzones in busy urban regions. The Delaunay triangulation does not have this drawback.

Krasnogor et al. (1995) and Lau and Shue (2001) are examples of papers in which the Delaunay triangulation is used in the same manner as in this chapter, i.e. to restrict the search space of the neighbourhood operators. For an example of a Delaunay triangulation see Figure 7.2.
The different moves that are used to explore the solution space are illustrated in Figure 7.3:

- The **2-opt** move removes two arcs, each arc between two natural neighbours, and reconnects the solution with two new arcs.

- The **swap** move exchanges two microzones that are natural neighbours *and* in different tours.

- The **relocate** move transfers a microzone from one tour to the position immediately after one of its natural neighbours in another tour. Natural neighbours inside the same tour are not regarded.

![Figure 7.3: Examples of the different moves used in the variable-neighbourhood tabu search algorithm](image)
7.4 Methodology

Algorithm 7.1 Diversification and storage sequence

1: Let $w_o \in [0, 1]$, be the weight for objective $o$, with step size $\Delta t = 0.1$
2: Initialization - calculate an initial solution given the assignment of microzones to vehicles of the tactical plan
3: for all combinations of $w_o \mid \sum_{o=1}^{n} w_o = 1$ do
4: Do variable neighbourhood tabu search
5: Store solution in the solution archive $A$
6: end for
7: Filter $A$
8: Return $A$

7.4.2 Variable neighbourhood tabu search

A general outline of our algorithm is presented in Algorithm 7.1. As mentioned, the aim of this chapter is not to develop a state-of-the-art multi-objective metaheuristic, but to provide a simple and transparent heuristic that can be used to illustrate the multi-objective vehicle routing problem.

The tactical plan is used as a starting point for the multi-objective variable neighbourhood tabu search in the sense that the initial solution is determined by assigning all microzones to their preferred vehicle and then calculating a travelling salesman solution per vehicle.

For each combination of weights, $w_o$, which sum up to 1 and which vary in steps of 0.1, an iteration of the variable neighbourhood tabu search is executed, starting from the initial solution. The varying weight combinations will guide the search in different directions, leading to a more diversified approximation of the Pareto set.
Chapter 7  Multi-objective microzone-based routing for courier companies

Algorithm 7.2 variable neighbourhood tabu search

1: Let $V$ be the solution vector
2: Let $N$ be the neighbourhood tracker, initiate $N = 0$
3: while $N < 3$ do
4:   Execute move $M_N$
5:   if no improvement found then
6:     Increase $N$
7:   else
8:     Store the current solution in $V$
9:   end if
10: end while
11: Find best solution in $V$ using weighted exponential sum of the objective values using $w_o$

The result retrieved from the variable neighbourhood tabu search routine is stored in a solution archive $A$, which is kept in memory until the end of the program. This solution archive is filtered before it is returned to the decision maker. The filtering step will remove all solutions from the archive that are dominated by another solution, only accepting non-dominated solutions.

Moves are executed using a “best-improvement” strategy. The tabu list is a list of microzones on which moves have been executed in the last $t$ iterations of the variable neighbourhood tabu search procedure, and each neighbourhood has its own tabu list. In a limited pilot experiment, it was established that a value of $t = 6$ robustly resulted in high-quality solutions.

The variable neighbourhood tabu search algorithm (see Algorithm 7.2) is executed for every weight combination that sums up to one, with
a step size of 0.1. A solution vector $V$ is created, and the algorithm keeps track of the neighbourhood it is exploring. Depending on the neighbourhood, a certain move is executed, and the respective tabu list is updated. If no improvement can be found, the algorithm moves on to the next neighbourhood. Otherwise, all solutions are evaluated using the weighted exponential sum (see Equation (7.7)) of the objective functions, and the best solution is used as the starting solution for the next iteration.

$$f(x) = \left[ w_1 f_1^n(x)^2 + w_2 f_2^n(x)^2 + w_3 f_3^n(x)^2 \right]^{\frac{1}{2}}$$

(7.7)

As it is not necessary, nor desirable to spend time capturing all Pareto-optimal solutions, $P$ is set to 2. This value for $P$ gives a curvature of the objective function, and can enable the algorithm to find more points on the Pareto front.

### 7.5 Computational results

In this section, a variable neighbourhood tabu search algorithm to solve the multi-objective vehicle routing problem, and balance a tactical plan with microzones, is proposed. The experiment is designed to investigate if there is a correlation between any of the three objective functions, and to attempt to find an explanation for this correlation. Further, the results of the experiment are used to investigate if the amount of deviation in workload from the expected workload in a microzone has a
large impact on the Pareto front found by the algorithm. A second experiment is conducted to examine if the variable neighbourhood tabu search algorithm is robust with respect to its initial solution.

### 7.5.1 Test instances

As stated before, the model assumes that a tactical plan is given, with known work times for the microzones and known microzone-vehicle allocations. To test the algorithm described in this chapter extensively, a set of realistic test instances is generated. The test instances comprise: zones that are assigned to vehicles in a tactical plan, an initial routing for the tactical plan, a limit in amount of work a vehicle (driver) can perform, the amount of work in the microzones and the coordinates of the microzones in the distribution zone.

To construct the instances, microzones are generated in randomly distributed locations on a square canvas (of size 1000 by 1000). To replicate city centres and industrial areas, or other areas with a denser customer base than rural areas, the instances are generated in such a way that 90% of the microzones are concentrated in clusters. The locations of the remaining 10% are uniformly generated.

To mimic the typical layout of a distribution area, microzone clusters are generated in such a way that the center of the cluster is denser than the boundaries. This models the fact that houses in city centres are built closer to each other, and apartment blocks are more common. To this end, the distance between a microzone and the center of the cluster
it belongs to is generated as an exponentially distributed variate with mean $\lambda^{-1}$, a parameter which can vary across different clusters.

To obtain a tactical plan from this generated instance, the open-source VRP solver, VRPH (Groer (2008)), is used to build routes using the microzones as customers and the distances between those microzones as travel times. These routes determine the tactical assignment of microzones to drivers.

In the generation of the tactical plan, the microzones are of equal workload. This is a realistic assumption as the generation of microzones for use in a tactical plan in a previous step in the “tactical plan–operational plan” approach, would mean that they are robust and as close as possible to equal workloads for known (historical) pick-up and delivery locations. Of course, for the daily operational plan the workload in each microzone can vary. For this reason, the operational demand in each microzone is generated as a single draw from a triangular distribution. The median of this triangular distribution is set equal to the tactical demand of the microzone, its minimum and maximum are set to $(100 - w)\%$ and $(100 + w)\%$ of the tactical demand respectively, where $w$ is allowed to vary between 0 and 30.

The instances used in this chapter, which can be found on http://antor.uantwerpen.be/ZVRP, use the following naming syntax:

$ZVRP_{<unique\ name>}_N<n=number\ of\ microzones>_n<n=number\ of\ clusters>_w<workload\ deviation>_M<maximum\ distance>_L<lambda-da>_case_<case\ number>_csv$
Table 7.1: Parameter values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Step size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of microzones (N)</td>
<td>100</td>
<td>200</td>
<td>50</td>
</tr>
<tr>
<td>Number of clusters (n)</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Maximum workload deviation (w)</td>
<td>0</td>
<td>30</td>
<td>5</td>
</tr>
<tr>
<td>Maximum distance (M)</td>
<td>1000</td>
<td>1000</td>
<td>0</td>
</tr>
<tr>
<td>Lambda (L)</td>
<td>20</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>Case number (case)</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

ZVRP refers to “zone-based vehicle routing problem”. The unique name is used to differentiate between instances where all other variables have identical values. The maximum distance refers to the size of the problem instance. The values we have used for the different parameters in our generated instances can be found in Table 7.1. An example of an instance name generated with a particular combination of values in the table and the unique name “full_test” would be:

ZVRP_full_test_N100_n0_w0_M1000_L20_case_0.csv

7.5.2 Results

The variable neighbourhood tabu search algorithm was run on 252 test instances, with number of zones varying from 100 to 200 in steps of 50, the number of clusters from 0 to 3 in steps of 1 and the maximum deviation in the work per zone from 0 to 30% in steps of 5%. Three different instances of all combinations of the parameters are created. Since this is a novel problem, no algorithms are available to compare
Figure 7.4: Pareto-front for an instance with 200 microzones and 1 cluster, for different values of maximum deviations of the amount of work in the microzones.
the results. We, therefore, discuss the results of our algorithm, which provide some insight into the structure of the problem. The average computational time needed for one instance is 4 minutes.

An example of a Pareto-front obtained using our method can be found in Figure 7.4. For an increase in maximum workload deviation there is no visible influence on the shape of the Pareto-front. In Table 7.2, we can see that the overall average percentage difference between the best hypervolume for an instance and the other hypervolumes for that instance is only 3%, with a maximum of only 5%. This shows that there are no big differences between the different Pareto sets we find for each instance. We can conclude that the maximum deviation in workload \( w_x \) for \( x \in [0, 30] \), with step size \( \Delta_t = 5 \) has nearly no impact on the final Pareto front obtained. Two possible explanations for this behaviour come to mind. The first one being that the workload imbalance in microzones gets resolved by a good combination of microzones (which would prove the usefulness of combining microzones for balancing workload between vehicles). The other possible explanation, that the amount of work inside a microzone is very small in comparison to the time spent driving between microzones and that the impact of different workloads inside the microzones is therefore very small, can be ruled out. The test instances have been generated to ensure that, on average, more than 40% of a vehicle’s time is spent inside the microzones.

The initial solution used by the algorithm has little influence on the Pareto front found. In Figure 7.5 the Pareto-fronts of the same problem instance, with different start solution and one tactical plan, are shown.
7.5 Computational results

<table>
<thead>
<tr>
<th></th>
<th>Average % difference</th>
<th>Maximum % difference</th>
<th>Standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>hypervol_N100_n0_case_0</td>
<td>1.39</td>
<td>4.56</td>
<td>1.47</td>
</tr>
<tr>
<td>hypervol_N100_n0_case_1</td>
<td>1.40</td>
<td>5.19</td>
<td>1.80</td>
</tr>
<tr>
<td>hypervol_N100_n0_case_2</td>
<td>0.66</td>
<td>1.48</td>
<td>0.55</td>
</tr>
<tr>
<td>hypervol_N100_n1_case_0</td>
<td>1.62</td>
<td>3.47</td>
<td>1.36</td>
</tr>
<tr>
<td>hypervol_N100_n1_case_1</td>
<td>2.81</td>
<td>4.33</td>
<td>1.48</td>
</tr>
<tr>
<td>hypervol_N100_n1_case_2</td>
<td>1.57</td>
<td>2.75</td>
<td>0.95</td>
</tr>
<tr>
<td>hypervol_N100_n2_case_0</td>
<td>1.27</td>
<td>2.30</td>
<td>0.77</td>
</tr>
<tr>
<td>hypervol_N100_n2_case_1</td>
<td>1.13</td>
<td>1.92</td>
<td>0.65</td>
</tr>
<tr>
<td>hypervol_N100_n2_case_2</td>
<td>0.68</td>
<td>1.60</td>
<td>0.60</td>
</tr>
<tr>
<td>hypervol_N100_n3_case_0</td>
<td>2.14</td>
<td>4.11</td>
<td>1.55</td>
</tr>
<tr>
<td>hypervol_N100_n3_case_1</td>
<td>1.03</td>
<td>1.60</td>
<td>0.58</td>
</tr>
<tr>
<td>hypervol_N100_n3_case_2</td>
<td>0.70</td>
<td>1.13</td>
<td>0.40</td>
</tr>
<tr>
<td>hypervol_N150_n0_case_0</td>
<td>0.53</td>
<td>1.09</td>
<td>0.41</td>
</tr>
<tr>
<td>hypervol_N150_n0_case_1</td>
<td>0.84</td>
<td>2.09</td>
<td>0.81</td>
</tr>
<tr>
<td>hypervol_N150_n0_case_2</td>
<td>1.44</td>
<td>2.30</td>
<td>0.73</td>
</tr>
<tr>
<td>hypervol_N150_n1_case_0</td>
<td>3.20</td>
<td>5.27</td>
<td>1.99</td>
</tr>
<tr>
<td>hypervol_N150_n1_case_1</td>
<td>0.89</td>
<td>2.37</td>
<td>0.76</td>
</tr>
<tr>
<td>hypervol_N150_n1_case_2</td>
<td>1.24</td>
<td>3.34</td>
<td>1.14</td>
</tr>
<tr>
<td>hypervol_N150_n2_case_0</td>
<td>0.78</td>
<td>1.12</td>
<td>0.37</td>
</tr>
<tr>
<td>hypervol_N150_n2_case_1</td>
<td>1.09</td>
<td>1.74</td>
<td>0.61</td>
</tr>
<tr>
<td>hypervol_N150_n2_case_2</td>
<td>1.30</td>
<td>3.27</td>
<td>1.08</td>
</tr>
<tr>
<td>hypervol_N150_n3_case_0</td>
<td>2.80</td>
<td>4.82</td>
<td>2.12</td>
</tr>
<tr>
<td>hypervol_N150_n3_case_1</td>
<td>1.21</td>
<td>1.91</td>
<td>0.67</td>
</tr>
<tr>
<td>hypervol_N150_n3_case_2</td>
<td>0.87</td>
<td>3.02</td>
<td>1.02</td>
</tr>
<tr>
<td>hypervol_N200_n0_case_0</td>
<td>1.87</td>
<td>3.89</td>
<td>1.27</td>
</tr>
<tr>
<td>hypervol_N200_n0_case_1</td>
<td>0.88</td>
<td>2.82</td>
<td>0.95</td>
</tr>
<tr>
<td>hypervol_N200_n0_case_2</td>
<td>0.97</td>
<td>1.61</td>
<td>0.52</td>
</tr>
<tr>
<td>hypervol_N200_n1_case_0</td>
<td>1.14</td>
<td>1.66</td>
<td>0.57</td>
</tr>
<tr>
<td>hypervol_N200_n1_case_1</td>
<td>1.18</td>
<td>2.01</td>
<td>0.70</td>
</tr>
<tr>
<td>hypervol_N200_n1_case_2</td>
<td>1.16</td>
<td>3.48</td>
<td>1.24</td>
</tr>
<tr>
<td>hypervol_N200_n2_case_0</td>
<td>0.73</td>
<td>1.44</td>
<td>0.55</td>
</tr>
<tr>
<td>hypervol_N200_n2_case_1</td>
<td>2.51</td>
<td>3.63</td>
<td>1.18</td>
</tr>
<tr>
<td>hypervol_N200_n2_case_2</td>
<td>0.53</td>
<td>1.75</td>
<td>0.63</td>
</tr>
<tr>
<td>hypervol_N200_n3_case_0</td>
<td>0.99</td>
<td>1.66</td>
<td>0.52</td>
</tr>
<tr>
<td>hypervol_N200_n3_case_1</td>
<td>0.32</td>
<td>0.80</td>
<td>0.32</td>
</tr>
<tr>
<td>hypervol_N200_n3_case_2</td>
<td>1.54</td>
<td>3.90</td>
<td>1.38</td>
</tr>
</tbody>
</table>
The figure presents a few of the Pareto-fronts found for the different start solutions, and demonstrates that they are generally very close to one another. In Table 7.3, the results are compared in terms of accuracy by using the hypervolume of the found Pareto sets. Eighty four different initial solutions have been tested. The average deviation from the best found Pareto set is 2.3%, showing that the method is relatively robust with respect to the initial solution.

Statistical tests on the data of the Pareto-fronts found by the algorithm reveal that there is a clear (inverse) correlation between the total driving distance and the deviation of the average workload (see Table 7.4 for an example). This result is intuitive, as the optimal solution for any instance with a vehicle that has no time or capacity constraint is the optimal TSP tour. If we have a fixed number of vehicles to bal-
7.5 Computational results

Table 7.3: Percentage difference in hypervolume with the best found Pareto set for different start solutions and one tactical plan

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Average difference</td>
<td>2.32</td>
</tr>
<tr>
<td>Maximum difference</td>
<td>15.0</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>2.41</td>
</tr>
</tbody>
</table>

ance the workload this tour should be split up in parts of roughly the same amount of work, and be assigned to the vehicles. Next, the travel from and to the depot should be added, which makes the total driving distance longer. The correlation between the deviation of the average workload and the deviation from the tactical plan is not always statistically significant, but logic dictates that there should be some correlation.

As we try to minimise the deviation of the average workload, it is obvious that a deviation of the operational solution from the tactical plan is inevitable. In deviating from this tactical plan, the driving distance becomes larger. If the total driving distance is minimised, the routes get more and more unbalanced, and we deviate to a greater extent from the tactical plan.

Table 7.4: Pearson’s product-moment correlation for driving distance and deviation of average worktime for instance with 200 microzones, 1 cluster and maximum workload deviation of 20%

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Values:</td>
<td>t = -22.559, df = 108, p-value &lt; 2.2e-16</td>
</tr>
<tr>
<td>Alternative hypothesis:</td>
<td>true correlation is not equal to 0</td>
</tr>
<tr>
<td>95% confidence interval:</td>
<td>-0.936 ; -0.869</td>
</tr>
<tr>
<td>Sample estimates:</td>
<td>cor -0.908</td>
</tr>
</tbody>
</table>
Because of the technique used to deal with the multi-objective property of the problem, it is not clear if this gives the true shape of the Pareto-optimal set, or just a convex approximation. The use of a different multi-objective optimisation method, or a true multi-objective metaheuristic could be helpful to give a better estimate of the real Pareto set.

Taking a look at the routes found by the algorithm proposed in this chapter (see e.g., Figure 7.6), however, it is clear that the decision maker should keep an eye on the routes she selects, and has to make a keen assessment of the routes proposed. If she pushes the inequality of the workload to the lowest point possible, and by doing so relaxes the constraint on driving distance, she has a high probability to find suboptimal routes. The algorithm will try to balance the routes in such a way that it does not regard the driving distance and hence will produce detours, just for the sake of balancing the routes. A healthy balance of the three objective functions should be maintained at all times. It is up to the decision maker to keep this balance, and selecting the optimal route configuration for her needs.

### 7.6 Conclusions and future research

In this chapter, we have described a multi-objective vehicle routing problem faced by several courier companies, and developed an algorithm to solve it. Afterwards, some results were shown and analysed. We demonstrated that a variable neighbourhood tabu search approach could be used to balance a tactical plan with microzones, but the de-
7.6 Conclusions and future research

Figure 7.6: Example solutions found for the instance with 150 zones, zero city centers and a deviation of workload in the zones of maximum 15% of the initial workload
cision maker must pay attention to the proposed results and assess its usability, and applicability on a case by case basis.

We find a clear (negative) correlation between total driving distance and deviation of average work time. Between total driving distance and deviation of the tactical plan we sometimes find a significant correlation, but this is not the case for all instances. The same holds for the correlation between deviation of the average work time and deviation from the tactical plan.

From the results of the experiment, we can conclude that the maximum deviation in workload \( w_x \) for \( x \in [0, 30] \), with step size \( \Delta t = 5 \) has nearly no impact on the final Pareto front obtained. Furthermore, we have shown that our approach is robust with respect to the initial solution.

A primary point of improvement is the speed of the algorithm. Currently all possible moves with nodes from neighbouring tours, tours which contain natural neighbours for the node that is checked, are considered. It might, for example, be possible to distinguish moves that have a higher probability to yield a good result from the ones that are rather unlikely to improve the solution, and decrease the computational time needed to evaluate the entire neighbourhood.

A second point of improvement would be the implementation of different ways to deal with the multi-objective character of the problem. In the current algorithm, a scalarization technique is used, which has the downside of not finding convex-dominated alternatives. More appropriate, but necessarily more elaborate alternatives to this technique
7.6 Conclusions and future research

could be considered, to make sure the Pareto set is not containing any non-convex parts, and to compare the results of that technique with the results of the scalarization technique. In (Caballero et al., 2007) all visited points are checked for inclusion in the list of Pareto-optimal points. Checking each solution visited can let the search sensibly increase the number of efficient solutions found while reducing the number of tabu searches and the computational effort.

The results obtained using different maximum working time deviations demonstrate that the algorithm is, to some extent, capable of combining zones in such a way that each vehicle has approximately the same amount of work.
Parameter tuning of a multi-objective algorithm using PROMETHEE

This chapter is based on the manuscript “Janssens, J., Sörensen, K., and Janssens, G. K. (2016b). A parameter tuning method to analyse the influence of algorithmic parameters in combination with instance characteristics on the quality of a Pareto front. Working paper, RPS-2016-006”
8.1 Introduction

Setting the parameters of an algorithm is not a trivial task. Many papers have been written about this subject. A novel approach to set the parameters of a multi-objective solution approach is proposed in this chapter. The approach proposed here, compares different Pareto sets obtained by the multi-objective variable neighbourhood tabu search proposed in Chapter 7.

There is no straightforward way of comparing different Pareto sets. The very definition of the quality of a Pareto set is not unambiguously defined. A plethora of papers are dedicated to the definition of the quality of a Pareto set, defining different indices to measure this quality. A classification of the indices has been done by Okabe et al. (2003), defining three main points of interest. The class of cardinality based indices, in which the number of solutions in the Pareto set and dominance of solutions from one Pareto set over an other Pareto set are bundled. Accuracy based indices that measure how close a Pareto set is to a theoretical Pareto front, or how much of the area of the solution space is dominated by a Pareto set. And finally, distribution and spread based indices, that measure how well and evenly spread the solutions of a Pareto set are in the solution space.

Both the parameters of the algorithm, as well as the instance characteristics might have an influence on the quality of the Pareto set, as defined by any of these classes of indices. Therefore, not only the parameters of the algorithm should be under study, but also the properties of the instances and the interaction between the properties of the instances and the algorithm parameters should be investigated.
We will start by doing a full factorial experiment that will execute the algorithm for different levels of the algorithm parameters on different instances (with different characteristics). This will return a Pareto set for each combination of parameter settings, for each instance. Three indices will be used to measure the quality of the Pareto sets, one for each of the three different classes of performance measures.

As we are comparing different Pareto sets to each other on different performance indices, this results in a multi-criteria decision problem. Therefore, the approach proposed here will make use of the PROMETHEE method (see Brans et al. (1986)) to make a ranking of the set of alternatives. The PROMETHEE method will return a net flow for each of the Pareto sets. We will do a statistical analysis (analysis of variance (ANOVA)) of the influence of the different parameter settings and their interaction effects with instance properties on this net flow and compare the results and conclusions from that analysis with the results from The PROMETHEE GDSS Procedure to verify our findings.

8.2 Literature

Much has been written about deciding which parameter values one should select to make their algorithms perform at their best. Hooker (1995) argues that researchers should focus on scientific testing rather than competitive testing. This would lead to a sound understanding of why certain algorithms work better than others, rather than just listing algorithms and their computational times. In Eiben and Smit
(2011), this leads to two different perspectives of parameter tuning: configuring an evolutionary algorithm by choosing parameter values that optimize its performance, and analysing an evolutionary algorithm by studying how its performance depends on its parameter values. In their paper, a conceptual framework for parameter tuning is presented. They also classify different tuning methods.

CALIBRE is proposed by Adenso-Diaz and Laguna (2006), and attempts to find the best values for up to five search parameters associated with a procedure under study. It performs a systematic search for parameter values within specified ranges employing a measure of performance as a guiding mechanism. CALIBRE makes use of Taguchi’s fractional factorial experimental designs, coupled with a local search procedure.

Instead of estimating the performance of an evolutionary algorithm for different parameter values or ranges of values, REVAC estimates the expected performance when parameter values are chosen from a probability density distribution $C$ with maximized Shannon entropy (Nannen and Eiben, 2007). This method is only appropriate for quantitative parameters (Eiben and Smit, 2011).

Racing is a technique that tests a set of models in parallel, quickly discards those models that are clearly inferior and concentrates the computational effort on differentiating among the better models (Maron and Moore, 1997).

F-Race is a racing algorithm that starts by considering a number of candidate parameter settings and eliminates inferior ones as soon as
enough statistical evidence arises against them (Balaprakash et al., 2007). An iterated F-Race is proposed in Birattari et al. (2010), and an implementation in R is presented in López-Ibáñez et al. (2011). López-Ibáñez and Stützle (2010) apply I/F-Race on a bi-objective TSP using both the hypervolume and the epsilon measures.

Sequential parameter optimization (SPO) (Bartz-Beielstein and Preuss, 2006; Bartz-Beielstein et al., 2004) is an iterative model based approach that entails two phases. First a primary model is built, which is afterwards evaluated and improved in a second step, that is executed iteratively.

Many of the aforementioned methods only deal with a single objective value, but could possibly be extended to cope with multi-objective optimisation problems. Furthermore, most of the methods mentioned before dealt with the solutions and their objective values directly rather than with quality of the Pareto set. The method proposed in this paper deals with indices that measure the quality of the Pareto set. To the best of our knowledge, only López-Ibáñez and Stützle (2010) and Janssens and Pangilinan (2010) took into account quality indices before.

Many real-world optimisation problems are multi-objective by nature and have objectives that are in conflict. Mathematical techniques are available to find best-compromise solutions by aggregating multiple objectives into a single function (Ehrgott, 2005). They have their drawbacks as they have difficulty dealing with concave and discontinuous Pareto fronts. Zitzler et al. (2004) stress the quality assessments of Pareto set approximations. In single-objective optimisation, quality can be defined by means of the objective function. While comparing
several solutions in the presence of multiple optimisation criteria, the concept of Pareto dominance can be used, but mostly leads to solutions being incomparable, i.e. neither of them dominates the others. It becomes more complicated when sets of solutions are compared as also most of the sets might be incomparable. Therefore it is not clear what ‘quality’ means with respect to Pareto set approximations.

Deb (2001) states that there are two orthogonal goals for any multi-objective optimisation algorithm: (1) to identify solutions as close as possible to the true Pareto-optimal set and (2) to identify a diverse set of solutions distributed evenly across the entire Pareto-optimal front. This has led to several metrics that characterise either closeness, or diversity, or both.

Three main classes of performance indices have been defined by Okabe et al. (2003), based on the performance indices’ properties: Cardinality based, accuracy based and distribution and spread based. We selected one from each of these classes to evaluate the Pareto sets we retrieved as output from the algorithm from Chapter 7. An example of a metric which measures the closeness to the Pareto surface is the Set coverage (Zitzler, 1999). To measure the diversity across the Pareto surface, the Schott’s Spacing (Schott, 1995) is presented. A measure like the hypervolume measures both closeness and diversity (Zitzler and Thiele, 1999). Most measures are unary quality measures, i.e. the measure assigns to each Pareto set approximation a number that reflects a certain quality aspect.
8.3 Metrics for the evaluation of the quality of Pareto sets

8.3.1 Set Coverage

Coverage ($C$), first proposed in Zitzler and Thiele (1998b), is used to compare two sets to each other in terms of dominance, and falls in the class of cardinality based performance indices. A solution vector $a$ is said to cover solution vector $b$ if $a$ weakly dominates $b$, notated as $a \geq b$. The coverage of set $S_1$ over set $S_2$ is given by:

$$C(S_1, S_2) = \frac{|\{s_2 \in S_2; \exists s_1 \in S_1 : s_1 \geq s_2\}|}{|S_2|}$$  \hspace{1cm} (8.1)

The reader should note that $C(S_1, S_2)$ does not have to be equal to $1 - C(S_2, S_1)$. The metric value $C(S_1, S_2) = 1$ means all members of $S_2$ are weakly dominated by $S_1$. On the other hand, $C(S_1, S_2) = 0$ means that no member of $S_2$ is weakly dominated by $S_1$.

As coverage is a pairwise metric, for every Pareto front we calculate the coverage with every other Pareto front, and average the coverages for each of them.

8.3.2 Hypervolume

As we lack the Pareto optimal set, for evaluating the quality of the obtained Pareto sets on accuracy, we can only compare them to each
other. For each Pareto set we calculate the objective value space covered by the set of non-dominated solution vectors, referred to as hypervolume (see Zitzler and Thiele (1998b)). The algorithm used in this chapter to calculate the hypervolume for a Pareto sets has been proposed by Fonseca et al. (2006). An example of a hyperarea, a special case of the hypervolume in two dimensions, is given in Figure 8.1.

![Figure 8.1: The hypervolume indicator in a two-objective case](image)

The hypervolume metric is interesting because it is a unary metric which is sensitive both to the overall advancement of the non-dominated set and to the distribution of individual points across the set. The placement of the reference point is critical and determines the sense and the magnitude of the hypervolume. Problems may appear if objectives have dissimilar scales or if some objectives are not bounded.
8.3.3 Spacing

To evaluate the distribution of the solution vectors in the retrieved Pareto sets, spacing (SP) is used. It was introduced by Schott (1995). Schott’s metric measures the diversity of a non-dominated set. It is calculated as

\[
SP(S) = \sqrt{\frac{1}{|S| - 1} \sum_{i=1}^{|S|} (d_i - \bar{d})^2}
\]  

(8.2)

\[
d_i = \min_{s_k \in S \land s_k \neq s_i} \sum_{m=1}^{M} |f_m(s_i) - f_m(s_k)|
\]  

(8.3)

where \( m \) is the number of objective functions and \( \bar{d} \) is the average of \( d_i \).

In Okabe et al. (2003) it is pointed out that this measure should be used with some caution, as a large gap between solution vectors can bias the results. A value of zero means that all solution vectors are equally spaced.

8.4 The Promethee method

To decide which Pareto set we prefer over the others, the Promethee method, introduced by Brans et al. (1986), is used in this paper. Promethee is an outranking method in multiple-criteria decision analysis (MCDA), that originally was proposed in two forms. Promethee
I, where a partial ranking is created, and PROMETHEE II, where a complete ranking is given as output. Two main parameters can be set by the decision maker to influence the behaviour of the ranking mechanism. For each criterion $f_i$, a weight $\pi_i$ which is a measure of relative importance of that criterion, can be given. Besides from the weight, a preference function for each criterion needs to be chosen. Six basic types of generalised criteria have been defined by Brans et al. and can be found in their paper. For some of those basic types of generalised criteria, extra parameters can be defined such as an indifference zone and a preference zone, for which the decision maker is indifferent or has a strict preference for one solution over the other.

### 8.4.1 PROMETHEE properties

We have chosen to keep the weights assigned to each index, used to measure the quality of the Pareto front, inside the PROMETHEE method the same. Varying these weights can favour one index over the others, but that is out of the scope of this work, and left to the decision maker. The outranking criterion chosen in this work is the fifth criterion proposed in Brans et al. (1986), the criterion with linear preference and indifference area. The values of $p$, which indicates the strict preference zone, and $q$, which is the maximum value for which the decision maker is indifferent between two alternatives, are set to the maximum difference between the values for each objective value and zero respectively. These values were taken to indicate that there is no strict preference and no indifference.

The preference function is a function of the difference between the
objective values for two alternatives (see Equation (8.4)). When $d$ is defined as $d = f(a) - f(b)$, Equation (8.5) defines the type five criterion used in this work.

$$P(a, b) = \mathcal{P}(f(a) - f(b)).$$ \hfill (8.4)

$$H(d) = \begin{cases} 
0 & \text{if } |d| \leq q, \\
(|d| - q)/(p - q) & \text{if } q < |d| \leq p, \\
1 & \text{if } p < |d|. 
\end{cases}$$ \hfill (8.5)

Figure 8.2: Linear preference and indifference area

In this chapter, two of the performance indices used are better when they are larger. Schott’s Spacing, however, is supposed to be as small as possible (but larger or equal to zero). The algorithm used in this chapter for the PROMETHEE method, is created with only maximisation

198
in mind. To handle the minimisation, all values of Schott’s Spacing performance index are multiplied by -1, so that it can be dealt with as a maximisation.

8.4.2 Promethee algorithm

The algorithm used to evaluate the alternatives starts by loading the instance and the Promethee parameters. For each instance it loads the set of alternatives ($\mathcal{A}$), the number of criteria ($k$), the weights ($\pi_i$), a value 1 for the criteria which should be maximised, and -1 otherwise ($b_i$), and values ($f_i(\mathcal{A})$) for each criteria for all the alternatives.

In the next step, the values for the different alternatives are multiplied with their respective $b_i$, to cope with a possible minimisation for a certain criterion. $f_i(\mathcal{A}) = f_i(\mathcal{A}) \cdot b_i$

Finally, for each alternative, an entering (Equation (8.6)) and leaving flow (Equation (8.7)) and its resulting net flow (Equation (8.8)) is calculated.

The entering flow is the sum of all the multi-criteria preference indices (II), of one alternative over all the other alternatives in set $\mathcal{A}$. The multi-criteria preference index II is defined as the weighted average of the preference functions $P_i$ as described in Equation (8.9). For more information on the calculation of the preference index and the positive and negative flows, the reader is referred to Brans et al. (1986) and Brans and Mareschal (2005).
\[ \phi^- (a) = \frac{1}{|\mathcal{A}| - 1} \sum_{b \in \mathcal{A}} \Pi(b, a) \quad (8.6) \]
\[ \phi^+ (a) = \frac{1}{|\mathcal{A}| - 1} \sum_{b \in \mathcal{A}} \Pi(a, b) \quad (8.7) \]
\[ \phi(a) = \phi^+ (a) - \phi^- (a) \quad (8.8) \]
\[ \Pi(a, b) = \frac{\sum_{i=1}^{k} \pi_i P_i (a, b)}{\sum_{i=1}^{k} \pi_i} \quad (8.9) \]

8.5 The PROMETHEE GDSS Procedure

For each instance that is examined in this chapter, the metaheuristic proposed in Chapter 7 is applied for the combinations of five different settings for the tabu list parameter and fifteen different options for the move order parameter, as explained in Section 8.7. The Pareto optimal sets, retrieved from the algorithm, are compared on three different performance indices by the PROMETHEE method, as described in Section 8.3 and Section 8.4. This gives us a list of net flows for each parameter combination for each instance.

The PROMETHEE Group Decision Support System Procedure (see Brans and Mareschal (2005)) is used as a mechanism to retrieve an overall ranking. This approach is intended to be used when one has to deal with multiple decision makers, but could be utilised to find a global “best” parameter setting as well.
Three phases are described in this technique, for which phase I and phase II are the generation of the alternatives and criteria, and the individual evaluation by each decision maker (using PROMETHEE) respectively.

The third phase entails the global evaluation which makes use of the individual evaluations, here the net flows for each instance, as criteria. Each of them is assigned a weight, which in this work is set equal to 1 for all instances. After that, the PROMETHEE method is applied to this matrix of alternatives, using the type 3 generalised criterion with strict preference $p = 2$. This is exactly the same as taking the type 5 generalised criterion with indifference $q = 0$ and strict preference $p = 2$.

### 8.6 Test instances

The model in Chapter 7 assumes that a tactical plan is given, with known work times for the microzones and known microzone-vehicle allocations. A set of realistic test instances is generated. The test instances comprise: zones that are assigned to vehicles in a tactical plan, an initial routing for the tactical plan, a limit on the amount of work a vehicle (driver) can perform, the amount of work in the microzones and the coordinates of the microzones in the distribution zone.

To construct the instances, microzones are generated in randomly distributed locations on a square canvas (of size 1000 by 1000). To replicate city centres and industrial areas, or other areas with a denser customer
base than rural areas, the instances are generated in such a way that 90% of the microzones are concentrated in clusters. The locations of the remaining 10% are uniformly generated. (This is only true if the option for clusters is larger than zero.)

To mimic the typical layout of a distribution area, microzone clusters are generated in such a way that the center of the cluster is denser than its boundaries. This models the fact that houses in city centres are built closer to each other, and apartment blocks are more common. To this end, the distance between a microzone and the center of the cluster, to which it belongs, is generated as a random variable following an exponential distribution with a parameter, \( \lambda^{-1} \), which is allowed to vary across different clusters.

To obtain a tactical plan from this generated instance, the open-source VRP solver, VRPH (Groer (2008)), is used to build routes using the microzones as customers and the distances between those microzones as travel times. These routes determine the tactical assignment of microzones to drivers.

In the generation of the tactical plan, the microzones are of equal workload. This is a realistic assumption as the generation of microzones for use in a tactical plan in a previous step in the “tactical plan–operational plan” approach, would mean that they are robust and as close as possible to equal workloads for known (historical) pick-up and delivery locations. Of course, for the daily operational plan the workload in each microzone can vary. For this reason, the operational demand in each microzone is generated as a single draw from a triangular distribution. The median of this triangular distribution is set equal to the
tactical demand of the microzone, its minimum and maximum are set to 
(100 − w)% and (100 + w)% of the tactical demand respectively, where 
w is allowed to vary between 0 and 30.

The instances used in this chapter, which can be found on http://antor.
uantwerpen.be/ZVRP. For an overview of the naming syntax for the 
instances and the range of values from which the instance properties 
are drawn, the reader is referred to Chapter 7. The instance character-
istics that are investigated are the number of microzones, the number 
of clusters and the deviation from the average workload.

8.7 Algorithmic parameter values

Two parameters of the algorithm that are investigated in this work are 
tabu list size and move order/move inclusion. The tabu list size ranges 
from a size of zero elements to eight elements. The parameter does not 
directly change the size in physical memory, but rather the number of 
iterations each element stays in the tabu list. On every iteration, one 
element is placed in the list for the specified amount of iterations. This 
directly relates to the number of elements that is present in the tabu 
list.

The second parameter determines the move order to be used in the 
variable neighbourhood tabu search. This move order defines which 
move is explored first, which moves will be explored next, if any, and 
in which order. The decision is made to have 3 consecutive moves at 
maximum, to keep the possible number of combinations to a manage-
able size. The move order does not necessarily comprise every move
defined in the chapter. All permutations without repetition of moves are examined. Each move is executed until no more improvement is found. This means that scheduling the same move twice in a row is not beneficial. The choice has been made to exclude any repetition of the moves, to reduce the number of combinations, even though executing some move \( x \) followed by move \( y \) followed by move \( x \) might yield a better result than only executing move \( x \) followed by move \( y \).

If all moves are included, six possible combinations exist. If only two moves are executed (one move is excluded), there are six more combinations. If only one move is used, three combinations are possible. This gives a total of 15 possible move orders, in comparison to the 81 \( (4^3 - 1 + 4^2 - 1 + 4^1 - 1) \) combinations, when the restriction on repetition is not used. An overview of the different move orders tested in this work is given in Table 8.1.

### 8.8 Analysis of the results

A full factorial experiment has been set up and executed, in which the algorithm described in Chapter 7 is run for every possible parameter combination (the tabu list size and move order as described in Section 8.7) on all instances, multiple times.

For each of the resulting Pareto fronts, the three evaluative metrics have been calculated. To remove any bias based on the characteristics of the instances (for example if an instance has all his microzones in very close working distance from each other, and an other has them wide spread, the one where they are close inherently has a lower total
8.8 Analysis of the results

<table>
<thead>
<tr>
<th>Move order ID</th>
<th>Order of moves</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2-opt – swap – relocate</td>
</tr>
<tr>
<td>1</td>
<td>2-opt – relocate – swap</td>
</tr>
<tr>
<td>2</td>
<td>swap – 2-opt – relocate</td>
</tr>
<tr>
<td>3</td>
<td>swap – relocate – 2-opt</td>
</tr>
<tr>
<td>4</td>
<td>relocate – 2-opt – swap</td>
</tr>
<tr>
<td>5</td>
<td>relocate – swap – 2-opt</td>
</tr>
<tr>
<td>6</td>
<td>2-opt – swap</td>
</tr>
<tr>
<td>7</td>
<td>2-opt – relocate</td>
</tr>
<tr>
<td>8</td>
<td>swap – 2-opt</td>
</tr>
<tr>
<td>9</td>
<td>swap – relocate</td>
</tr>
<tr>
<td>10</td>
<td>relocate – 2-opt</td>
</tr>
<tr>
<td>11</td>
<td>relocate – swap</td>
</tr>
<tr>
<td>12</td>
<td>2-opt</td>
</tr>
<tr>
<td>13</td>
<td>swap</td>
</tr>
<tr>
<td>14</td>
<td>relocate</td>
</tr>
</tbody>
</table>

On the resulting performance indices, the Promethee method is applied on a per instance per run (of the full factorial experiment) basis. The net flows retrieved from Promethee are analysed statistically with ANOVA to identify which parameters or instance characteristics have a significant impact on the resulting net flows.

The statistical analysis starts by fitting a model with up to third order interactions, which returns a model with a reasonable fit and a large adjusted $R^2 = 0.925$, but the high number of large ($> 0.10$)
p-values makes it clear that the model has many insignificant terms (see NIST/SEMATECH (2013)). Starting from this model, a stepwise regression was performed based on AIC to eliminate unnecessary terms. By a combination of stepwise regression and the removal of remaining terms with a p-value larger than 0.05, a model with an intercept, nine significant effect terms and six non-significant effect terms with an adjusted $R^2 = 0.928$ was found. The main effects that are not significant on their own, but are part of a significant interaction effect were left in the model. The decision was made to only analyse main effects and second order interaction effects. The main effects move order and tabu list size and second order interaction effects Number of microzones:move order, number of clusters:move order, number of clusters:tabu list size and workload deviation:move order are further investigated. The distribution of the residuals is shown in Figure 8.3.

After examining the Tukey HSD for the parameter tabu list size ($t$), which is plotted in Figure 8.4, the conclusion can be made that there is a significant difference between tabu list size equal to eight and every other tabu list size, and between tabu list size six and zero. If the box plot of the tabu list size is examined, verification can be made that the mean net flow for tabu list size eight is slightly lower than the the other mean values. The plots of the interaction effect between tabu list size and number of clusters is shown in Figure 8.5 and Figure 8.6.

For a higher number of clusters, a larger tabu list size seems to become more important. Further experiments are needed to show if this trend holds true for more than 2 clusters.

For the other interaction effects, the move order does seem to have a
8.8 Analysis of the results

Figure 8.3: Residual plot for the fitted model

Figure 8.4: Tukey HSD 95% confidence intervals for tabu list size
significant impact. The same three move orders are being ranked the highest consistently, and from the Tukey HSD, we can conclude that no significant difference exists between the mean net flow for these move orders. Move orders 4, 5 and 11 take the highest ranking places for workload deviation, number of clusters and number of microzones. (see Figure 8.7, 8.8 and 8.9)

Move order 1 always ranks in the top 5. These move orders seem to have in common that the relocate move is directly or indirectly followed by the swap move. The presence and order of the 2-opt move seems to be of less importance. Further experiments should indicate if this effect is reproducible, and what is causing the 2-opt move to be of less importance to the quality of the Pareto front. The poorest performing move order seems to be number 13, which is the swap move. No
Figure 8.6: Interaction plot of tabu list size and number of clusters

Figure 8.7: Interaction plot of number of clusters and move order
Chapter 8 Tuning multi-objective algorithm parameters using PROMETHEE

Figure 8.8: Interaction plot of number of microzones and move order

Figure 8.9: Interaction plot of workload deviation and move order
other clear trend seems to be present in the interaction plots.

The Promethee GDSS procedure was also run on the Promethee net flows. The top 20 ranked alternatives are shown in Table 8.2. The results found by the GDSS procedure are consistent with our analysis with ANOVA. The highest ranking move orders are 1, 4, 5 and 11, and tabu list sizes 0, 2, 4 and 6 have a slight edge on tabu list size 8. Of course this is a general ranking, and specific instance properties might give a different ranking as described above.

![Box plot of move order](image)

Figure 8.10: Box plot of move order
Table 8.2: Top 20 ranking net flows for the GDSS procedure

<table>
<thead>
<tr>
<th>Move order</th>
<th>Tabu list size</th>
<th>PROMETHEE net flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>2</td>
<td>0.0828</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0.0821</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0.0818</td>
</tr>
<tr>
<td>11</td>
<td>4</td>
<td>0.0809</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0.0802</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0.0799</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0.0798</td>
</tr>
<tr>
<td>11</td>
<td>6</td>
<td>0.0788</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>0.0785</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>0.0785</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>0.0765</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>0.0758</td>
</tr>
<tr>
<td>11</td>
<td>8</td>
<td>0.0738</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>0.0723</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>0.0718</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>0.0641</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>0.0633</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0.063</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>0.0624</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0.061</td>
</tr>
</tbody>
</table>

8.9 Conclusions

To identify which parameter settings and which instance characteristics influenced the quality of the Pareto fronts, and how they do this, in this chapter, the Pareto fronts from the algorithm which is introduced in the previous chapter have been analysed. To obtain those
Pareto fronts, a full factorial experiment was executed. Performance indices for the resulting Pareto fronts were calculated and passed to the PROMETHEE method to retrieve a ranking per instance per run of the experiment. This ranking was then analysed with ANOVA.

From the statistical experiments conducted in this chapter, it is shown that the move order has an important impact on the ranking of the Pareto fronts in the PROMETHEE method. This indicates that the move order is largely responsible for the quality of the Pareto front found by the algorithm proposed in Chapter 7. Four instances of move order where consistently ranked in the top 5 of highest mean net flow for the different characteristics of the test instances. This indicates that even if the specifications of the instances change, these move orders seem to deliver high quality solutions and Pareto fronts. The swap move alone seems to perform poorly in terms of Pareto front quality, and gets ranked very low. No further important effects could be found by visually inspecting the interaction plots for move order and instance characteristics.

Secondly, the tabu list size of 8 in first instance seemed to have a bad impact on the quality of the Pareto front as it has a lower mean net flow than the rest of the tabu list sizes. But more detailed investigation of the interaction effect with the number of clusters showed that for a larger number of clusters (in the case of this experiment 2), the tabu list size of 8 found a better quality Pareto front than the other tabu list sizes. As a matter of fact, the order of the ranking had seemed to be reversed. Further experiments should validate this result and should clarify if this effect holds true for more clusters, and larger sizes of the tabu list.
Conclusions
In this dissertation, the results of the author’s doctoral research are presented. All research is aimed at facilitating decision makers with models and algorithms to help them in different stages of their planning process. Ranging from increasing security and safety in utility networks in different stages of the network life time, to guiding the operational planning stage in a courier company. A broad industrial range of problems is covered in this manuscript.

In the first part of this work, different planning stages of utility networks are investigated, taking special interest in various aspects and methods to increase security and safety in them. An efficient variable neighbourhood tabu search algorithm is designed to solve a budget allocation problem proposed in Chapter 2 and its extended version in Chapter 3. It successfully allocates a limited security budget among different security strategies to increase security in an existing network infrastructure.

The variable neighbourhood tabu search heuristic can find very good results in a really short amount of time. It is performing extremely well in comparison to the naive exact approach. Both chapters try to clarify why the algorithm works well and which parameters of the algorithm are helping it to find good results. Two alternative approaches to calculate and evaluate the network reliability are proposed.

The ideas and knowledge gained from Chapter 2 and Chapter 3 helped me in developing an iterated local search algorithm that is applicable to the problem where the decision maker is faced with increasing the total failure time associated to a domino event within a chemical plant, as shown in Chapter 4. This is possible by investing a limited budget in
Conclusions

safety barriers which may delay the propagation of the accident within a plant.

Differently from a myopic optimisation, where barriers are allocated to simply prevent a primary domino accidents, the model proposed in this chapter can be used to analyse more realistic scenarios in which higher order domino effects are considered. The solution approach, developed, not only is able to determine the ideal allocation of safety barriers to increase the intervention time in case of a domino accident of a given cardinality associated to the worst case scenario, but it can provide useful information of the maximum intervention times to stop the escalation depending on the installation where the domino accident has originated.

A final aspect of increasing security and safety in utility networks was tackled in Chapter 5. A mathematical model of the water distribution network design problem where the goal is to optimise the reliability in the network given a limited budget is proposed. An Adaptive Large Neighbourhood search solution method with different destroy and repair neighbourhoods is proposed and compared to a naive exact approach to solve said model.

The analysis shows that the algorithm is very effective. The most influential parameter was the number of iterations the algorithm could run. The most powerful part of the algorithm was the CleanUpSolution method. This method made the difference between finding the optimal solution or not. Future research should point out if specific destroy or repair neighbourhoods had a significant impact on the result.
In the second part of this dissertation, a model and algorithm that helps a decision maker in creating an operational planning for a courier company, starting from a tactical plan with microzones, is proposed. The problem that was solved with the proposed algorithm is of a multi-objective nature. The relationship between these objectives is investigated. A clear (negative) correlation between total driving distance and deviation of average work time is found. The maximum deviation in workload has nearly no impact on the final Pareto front obtained. Furthermore, the proposed approach is proven to be robust with respect to the initial solution.

To identify which parameter settings and which instance characteristics influenced the quality of the Pareto fronts, and how they do this, in Chapter 8, the Pareto fronts from the algorithm of Chapter 7 have been analysed. A range of conclusions have been made by analysing the statistical results with ANOVA.

In each of the chapters, future research goals have been defined. The consensus is that models can be extended to facilitate more realistic cases, but the results and algorithms look promising. The impact of different routines and algorithm properties on the quality of the solutions should get further attention in the future.
Bibliography


Bibliography


226


Bibliography


Petroleum Safety Authority Norway (2014). Regulations relating to management and the duty to provide information in the petroleum activities and at certain onshore facilities (the management regulations).


Sklet, S. (2006). Safety barriers: Definition, classification, and perform-
Bibliography


Winter, B. (2013). Linear models and linear mixed effects models in R with linguistic applications. *CoRR, abs/1308.5499*.


## Notation

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{G}$</td>
<td>a graph</td>
</tr>
<tr>
<td>$\mathcal{N}$</td>
<td>the set of nodes in graph $\mathcal{G}$</td>
</tr>
<tr>
<td>$\mathcal{E}$</td>
<td>the set of edges connecting the nodes in $\mathcal{N}$</td>
</tr>
<tr>
<td>$p_i$</td>
<td>probability of failure of the $i$-th edge</td>
</tr>
<tr>
<td>$S_i$</td>
<td>set of available security strategies of the $i$-th edge</td>
</tr>
<tr>
<td>$c_{ij}$</td>
<td>cost of the $j$-th security strategy of the $i$-th edge</td>
</tr>
<tr>
<td>$p_{ij}$</td>
<td>probability of failure of the $i$-th edge, given that the $j$-th security strategy is applied</td>
</tr>
<tr>
<td>$c_{i0}$</td>
<td>“do-nothing” strategy for the $i$-th edge</td>
</tr>
<tr>
<td>$o$</td>
<td>origin node</td>
</tr>
<tr>
<td>$d$</td>
<td>destination node</td>
</tr>
<tr>
<td>$\mathcal{E}_l^F$</td>
<td>set of edges that fail in scenario $l$</td>
</tr>
<tr>
<td>$\mathcal{E}_l^N$</td>
<td>set of edges that do not fail in scenario $l$</td>
</tr>
<tr>
<td>$C$</td>
<td>set of critical scenarios</td>
</tr>
<tr>
<td>$R_l$</td>
<td>probability of occurrence of scenario $l$</td>
</tr>
<tr>
<td>$x_{ij}$</td>
<td>decision variable that takes value 1 when security strategy $j$ is applied to edge $i$</td>
</tr>
<tr>
<td>$B$</td>
<td>the available security budget</td>
</tr>
<tr>
<td>$N_k$</td>
<td>the $k$-th neighbourhood</td>
</tr>
</tbody>
</table>
Notation

$\bar{E}$: set of edges that have a security strategy applied to them

$p_{i,j}^{e}p_{k}^{n}$: probability of failure of the $i$-th edge, $k$-th node

$S_{i}^{e},S_{k}^{n}$: set of available security strategies of the $i$-th edge, $k$-th node

$c_{i,j},c_{k,j}^{n}$: cost of the $j$-th security strategy of the $i$-th edge, $k$-th node

$p_{i,j}^{e},p_{k}^{n}$: probability of failure of the $i$-th edge, $k$-th node, given that the $j$-th security strategy is applied

$c_{i0},c_{k0}^{n}$: “do-nothing” strategy for the $i$-th edge, $k$-th node

$\mathcal{E}_{l}^{F}$: set of edges that fail in scenario $l$

$\mathcal{E}_{l}^{N}$: set of edges that do not fail in scenario $l$

$\mathcal{N}_{l}^{F}$: set of nodes that fail in scenario $l$

$\mathcal{N}_{l}^{N}$: set of nodes that do not fail in scenario $l$

$x_{i,j},x_{k,j}^{n}$: decision variable that takes value 1 when security strategy $j$ is applied to edge $i$, node $k$

$Con_{l}$: index of connectivity for scenario $l$

$r_{o,d}^{l}$: indicates if a functional path exists between node $o$ and node $d$ under scenario $l$

$\rho_{i,j}^{e},\rho_{k}^{n}$: potential of edge $i$, node $k$

$f_{i}^{e},f_{k}^{n}$: indicates if edge $i$ is present in set $\mathcal{E}_{l}^{F}$, node $k$ is present in set $\mathcal{N}_{l}^{F}$

$\mathcal{F}$: set of edges used in the spanning tree

$\mathcal{A}$: set of edges not used in the spanning tree

$\mathcal{K}$: set of critical nodes

$S_{c}$: set of scenarios that is critical for critical node $c$

$x_{a}$: indicates if edge $a$ is added to the solution

$c_{a}$: cost of adding edge $a$ to the solution
\( \mathcal{X} \) set of edges, for which the edges are in \( \mathcal{A} \) and which are added to the solution

\( Q \) set of edges which are in \( \mathcal{F} \) or in \( \mathcal{X} \)

\( Q^F_l \) subset of edges of \( Q \) that fail in critical scenario \( l \)

\( Q^N_l \) subset of edges of \( Q \) that do not fail in critical scenario \( l \)

\( P_c \) calculated allowed probability of failure allowed at critical node \( c \)

\( M_c \) maximum allowed probability of failure allowed at critical node \( c \)

\( \Delta_e \) difference between the the objective value of the previous and the current iteration

\( T \) temperature of the acceptance function

\( T_\alpha \) predefined adjustment value for the temperature

\( \delta \) decay value

\( \rho_d^- \) set probabilities of the destroy neighbourhoods

\( \rho_r^+ \) set probabilities of the repair neighbourhoods

\( D \) cardinality

\( pt_{ij} \) propagation time for edge \( (i, j) \)

\( \mathcal{M}_{ij} \) set of protective measures available for edge \( (i, j) \)

\( c_{ij}^k \) cost of protective measure \( k \) of edge \( (i, j) \)

\( e_{ij}^k \) effectiveness of protective measure \( k \) of edge \( (i, j) \)

\( \mathcal{F}^D_i \) set of fire-paths of cardinality \( D \) for node \( i \)

\( p_k \) fire-path \( k \)

\( PT_{ij} \) propagation time of the fire along edge \( (i, j) \)

\( ET_i \) escalation time after which a domino effect of cardinality \( D \) is initiated as a consequence of an incident at node \( i \)
Notation

\( x_{ij}^k \) decision variable that value takes one when protective measure \( k \) is applied to edge \((i, j)\), zero otherwise

\( \mathcal{Z} \) set of microzones

\( \mathcal{V} \) set of vehicles

\( R \) number of available vehicles

\( y_{ik} \) variable that takes value one if microzone \( i \) is assigned to vehicle \( k \) in the tactical plan

\( t_i \) time needed to deliver all parcels inside microzone \( i \)

\( t_{ij} \) time it takes a vehicle to drive between \( i \) and \( j \)

\( y_{ik} \) decision variable that takes value one when microzone \( i \) is assigned to vehicle \( k \) in the operational plan

\( x_{ij} \) decision variable that takes value one when the assigned vehicle travels directly between microzone \( i \) and \( j \)

\( l_k \) workload of vehicle \( k \)

\( W \) with a maximal allowed working time per driver

\( w_o \) weight of objective function \( f_{o,n} \)
Acronyms

**ALNS**  adaptive large neighbourhood search

**ANOVA**  analysis of variance

**BLEVE**  boiling liquid expanding vapour explosion

**GRASP**  greedy randomized adaptive search procedure

**ILS**  iterated local search

**LSNDP**  link-survivable network design problem

**MOTS**  Multi-Objective Tabu Search

**NSNDP**  node-survivable network design problem

**RCL**  restricted candidate list

**VCE**  vapour cloud explosion

**VND**  variable neighbourhood descent

**VNS**  variable neighbourhood search

**VNTS**  Variable neighbourhood tabu search

**VRP**  vehicle routing problem
Index

adaptive large neighbourhood search, see ALNS
ALNS, 129

breadth-first search, 68

CALIBRE, 191
cardinality, 88
combination, see scenario
critical, 59
critical node, 116, 118
criticality, 31, 60
Delaunay triangulation, 36, 168
domino effect, 83
event, see failure
exact approach, 42, 64, 123
failure, 22

fire-path, 91, 97
graph, 18, 57, 90, 100, 116, 163
GRASP, 30, 67, 97
greedy random adaptive search procedure, see GRASP

hypervolume, 162, 195

ILS, 65, 98
index of connectivity, 68
iterated local search, see ILS
knapsack problem, 95
lexicographic order, 94
local search, 29, 71
microzone, 158
minimal cut set, 47
Notation

multi-criteria preference index, 199
neighbourhood, 33, 71, 97, 173
destroy, 133
repair, 133
network, see graph
robust, 13
node group, 126, 127
operational route plan, 164

Promethee, 190, 196
Pareto-optimal set, 161
path or no path problem, 20
perturbation, 35, 72, 98
potential, 67, 68, 70
probability, 16, 23
  of disconnection, 23
  of failure, 19, 57, 118
quality indices
  accuracy based, 189
  cardinality based, 189
  distribution and spread based, 189
racing, 191
  F-Race, 191
  iterated F-Race, 192
RCL, 31, 67
redundant
  level-one, 121
  level-three, 122
restricted candidate list, see RCL
REVAC, 191
risk, 20
  assessment, 14, 16
safety, 5
safety barrier, 85, 101
scenario, 21, 22, 59
  critical, 21, 117, 127
Schott’s spacing, 196
security, 5
security strategy, 11, 20, 57
selection mechanism
  best improvement, 32, 68
  first improvement, 32, 68
  fitness proportion, see roulette wheel
  roulette wheel, 131
sequential parameter optimization, 192
set coverage, 194
simulated annealing, 133
survivable networks, 12
<table>
<thead>
<tr>
<th>Term</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>link-survivable</td>
<td>118</td>
</tr>
<tr>
<td>node-survivable</td>
<td>119</td>
</tr>
<tr>
<td>tabu list</td>
<td>29, 32, 35, 65, 97, 172</td>
</tr>
<tr>
<td>update formula</td>
<td>63</td>
</tr>
<tr>
<td>variable neighbourhood descent</td>
<td>see VND</td>
</tr>
<tr>
<td>variable neighbourhood search</td>
<td>see VNS</td>
</tr>
<tr>
<td>VND</td>
<td>33, 71, 97</td>
</tr>
<tr>
<td>VNS</td>
<td>33</td>
</tr>
</tbody>
</table>